(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 1 231 532 A2

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
14.08.2002 Bulletin 2002/33

(51) Int Cl.[7]: **G06F 1/00**

(21) Application number: **02250898.0**

(22) Date of filing: **08.02.2002**

(84) Designated Contracting States:
AT BE CH CY DE DK ES FI FR GB GR IE IT LI LU MC NL PT SE TR
Designated Extension States:
AL LT LV MK RO SI

(30) Priority: **09.02.2001 JP 2001033114**
29.03.2001 JP 2001094803

(71) Applicant: **SONY CORPORATION**
Tokyo 141 (JP)

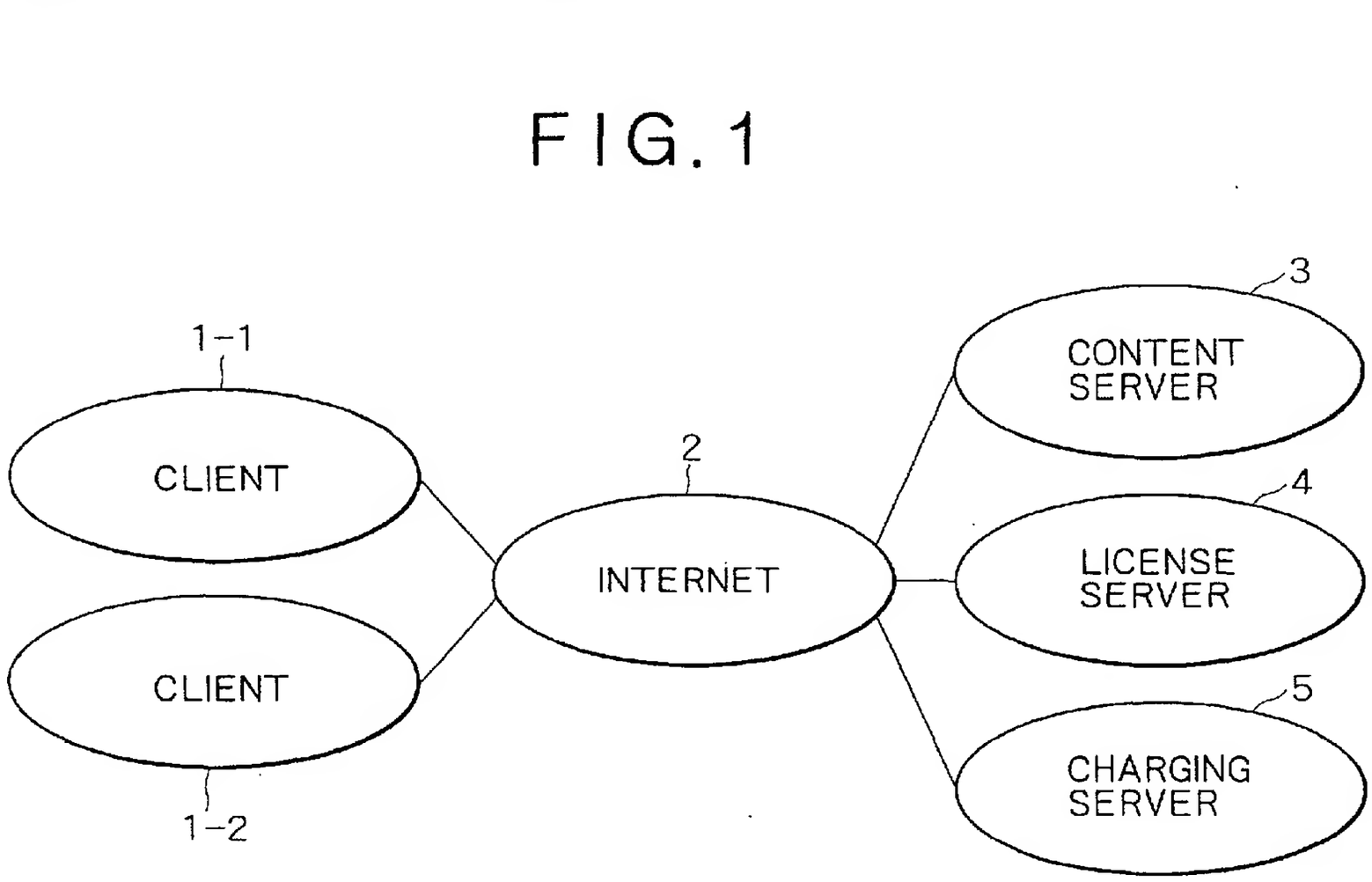
(72) Inventors:
- **Tanaka, Koichi, c/o Intellectual Property Dep.**
Shinagawa-ku, Tokyo 141 (JP)
- **Kawakami, Itaru, c/o Intellectual Property Dep.**
Shinagawa-ku, Tokyo 141 (JP)
- **Kuroda, Yoshisuke,**
c/o Intellectual Property Dep.
Shinagawa-ku, Tokyo 141 (JP)
- **Ishiguro, Ryuji, c/o Intellectual Property Dep.**
Shinagawa-ku, Tokyo 141 (JP)

(74) Representative: **Pratt, Richard Wilson et al**
D. Young & Co,
21 New Fetter Lane
London EC4A 1DA (GB)

(54) Information processing system for licensing content

(57) A client (1-1,1-2) receives an encrypted content from a content server (3). The header of the content includes license-identifying information for identifying a license required in utilization of the content. The client requests a license server (4) to transmit a license identified by the license-identifying information. When receiving the request for a license, the license server (4) carries out a charging process before transmitting the license to the client(1-1). The client stores the license received from the license server. The stored license serves as a condition for encrypting and playing back the content. As a result, a content can be distributed with a high degree of freedom and only an authorized user is capable of utilizing the content.

FIG. 1



EP 1 231 532 A2

Printed by Jouve, 75001 PARIS (FR)

Description

BACKGROUND OF THE INVENTION

[0001] In general, the present invention relates to an information-processing method, an information-processing apparatus, a program storage medium, a license server and a program.

[0002] Embodiments of the present invention relate to an information-processing method and an information-processing apparatus, which are used for preventing a content from being copied and used illegally without a license from the owner of the content copyright, a program for implementing the information-processing method and a program storage medium for storing the program.

[0003] In recent years, a user provides musical data owned by itself to another user and receives musical data not owned by itself from another user through the Internet in a content-exchanging system that allows a plurality of users to exchange musical data for free of charge.

[0004] In such a content-exchanging system, theoretically, music or another content owned by one user can thus be enjoyed by other users. Therefore, many users do not have to purchase such a piece of music and such a content. As a result, since such a piece of music or such a content is not well sold, the owner of the content copyright loses an opportunity to gain a royalty for the use of the piece of music or the content accompanying the sales of the piece of music or the content.

[0005] In society, there is thus a demand for prevention of a content from being copied and used illegally.

SUMMARY OF THE INVENTION

[0006] Embodiments of the present invention seek to address the problems described above to reliably prevent a content from being used illegally.

[0007] In accordance with an aspect of the present invention, there is provided an information-processing apparatus for allowing usage of a content by requiring a license for using the content. The information-processing apparatus includes:

content storage means for storing a license-specifying information for specifying the license required by the user in using the content, encrypted data of the content and key information required for decrypting the encrypted data of the content;
license storage means for storing the license including a content-specifying information for specifying the content, the use of which is allowed;
judgment means for forming a judgment as to whether or not the license required by the user in using the content has been stored in the license storage means; and
decryption means, which is used for decrypting the encrypted data of the content on condition that an outcome of the judgment formed by the judgment means indicates that the license required by the user in using the content has been stored in the license storage means.

[0008] The information-processing apparatus further includes transmission means for transmitting a license request including the license-identification information for identifying the license required by the user in using the content and receiving means for receiving the license transmitted by a license server. In addition, the license received by the receiving means is stored in the license storage means.

[0009] The information-processing apparatus further includes reproducing means for reproducing the content's data decrypted by the decryption means, wherein the data of the content is text data, image data, audio data, moving-picture data or a combination of them.

[0010] The information-processing apparatus further includes device-node-key storage means for storing a device node key. The key information includes an EKB (Enabling Key Block). The decryption means decrypts the EKB (Enabling Key Block) by using the device node key stored in the device-node-key storage means, and decrypts data of the content by using a root key obtained as a result of decryption of the EKB.

[0011] In the information-processing apparatus, the key information further includes a content key encrypted by using the root key of the EKB (Enabling Key Block). The data of the content is encrypted by using the content key. The decryption means decrypts the data of the content encrypted by using the content key by using the root key obtained as a result of decryption of the EKB (Enabling Key Block) by using the device node key stored in the device-node-key storage means.

[0012] In the information-processing apparatus, the license further includes usage-condition information showing a condition for using the content, use of which is allowed by the license.

[0013] In the information-processing apparatus, the license further includes an electronic signature signed by using a secret key of a license server.

[0014] The information-processing apparatus further has terminal-ID storage means for storing a terminal-specifying information identifying the information-processing apparatus. The license request transmitted by the transmission means further includes the terminal ID stored in the terminal-ID storage means. The license received by the receiving means includes a terminal ID. The judgment means compares the terminal-identification information included in the license with the terminal-identification information stored in the terminal-ID storage means, and determines that the license received by the receiving means is a license allowing use of the content only if the terminal ID included in the license matches the terminal-identification information stored in the terminal-ID storage means.

[0015] In accordance with another aspect of the present invention, there is provided an information-processing method for allowing a user to use a content by requiring the user to have a license for using the content. The information-processing method comprises:

a content storage step of storing a license ID for specifying the license required by the user in using the content, encrypted data of the content and key information required for decrypting the encrypted data of the content;
a license storage step of storing the license including a content-specifying information for specifying the content, the use of which is allowed by the license;
a judgment step of forming a judgment as to whether or not the license required by the user in using the content has been stored in the license storage means; and
a decryption step, at which the encrypted data of the content is decrypted on condition that an outcome of the judgment formed at the judgment means indicates that the license required by the user in using the content has been stored in the license storage means.

[0016] In accordance with a further aspect of the present invention, there is provided a program to be executed by a computer for carrying out processing of allowing a user to use a content by requiring the user to have a license for using the content. The program comprises:

a content storage step of storing a license specifying information for specifying the license required by the user in using the content, encrypted data of the content and key information required for decrypting the encrypted data of the content;
a license storage step of storing the license including a content specifying information for specifying the content, the use of which is allowed by the license;
a judgment step of forming a judgment as to whether or not the license required by the user in using the content has been stored in the license storage means; and
a decryption step, at which the encrypted data of the content is decrypted on condition that an outcome of the judgment formed at the judgment means indicates that the license required by the user in using the content has been stored in the license storage means.

[0017] The program or a portion of the program is encrypted.

[0018] In accordance with a still further aspect of the present invention, there is provided a license server for issuing a license for allowing use of a content. The license server comprises:

license storage means for storing the license including:

content-specifying information for specifying the content, use of which is allowed by the license; and
terminal-identification information for identifying an information-processing apparatus;
receiving means for receiving a license request including a license-identification information for identifying the license from the information-processing apparatus;
extraction means for extracting the license identified by the license-identification information included in the license request from the license storage means;
processing means for adding the terminal-identification information to the license extracted by the extraction means;
signature means for putting a signature on the license including the terminal-identification information added by the processing means by using a secret key of the license server; and
transmission means for transmitting the license with the signature put thereon by the signature means to the information-processing apparatus, from which the license request was received.

[0019] In accordance with a still further aspect of the present invention, there is provided another information-processing method for issuing a license for allowing use of a content. The other information-processing method comprises:

a license storage step of storing the license including a content-specifying information for specifying the content, use of which is allowed by the license and a terminal-identification information for identifying an information-processing apparatus;
a receiving step of receiving a license request including a license-identification information for identifying the license from the information-processing apparatus;
an extraction step of extracting the license stored in the license storage means and identified by the license-identification information included in the license request;
a processing step of adding the terminal-identification information to the license extracted at the extraction step;
a signature step of putting a signature on the license including the terminal-identification information added at the processing step by using a secret key of a license server used in the information-processing method; and

a transmission step of transmitting the license with the signature put thereon at the signature step to the information-processing apparatus, from which the license request was received.

[0020] In accordance with a still further aspect of the present invention, there is provided another program to be executed by a computer for carrying out processing of issuing a license for allowing use of a content. The other program comprises:

a license storage step of storing the license including a content-identification information for specifying the content, use of which is allowed by the license and a terminal-identification information for identifying an information-processing apparatus;
a receiving step of receiving a license request including a license-identification information for identifying the license from the information-processing apparatus;
an extraction step of extracting the license stored at the license storage step and identified by the license-identification information included in the license request;
a processing step of adding the terminal-identification information to the license extracted at the extraction step;
a signature step of putting a signature on the license including the terminal-identification information added at the processing step by using a secret key used in the information-processing method; and
a transmission step of transmitting the license with the signature put thereon at the signature step to the information-processing apparatus, from which the license request was received.

[0021] In the information-processing method, the information-processing apparatus and the program, which are provided by the present invention, a content is decrypted and can be used on condition that the user has a license for using the content.

[0022] In the license server and the information-processing method provided by the present invention, a valid license is issued only to a specific information-processing apparatus.

BRIEF DESCRIPTION OF THE DRAWINGS

[0023] For a better understanding of the present invention, reference will now be made by way of example to the accompanying drawings in which:

Fig. 1 is a block diagram showing the configuration of the context-exchanging system to which an embodiment of the present invention is applied;
Fig. 2 is a block diagram showing the configuration of a client shown in Fig. 1;
Fig. 3 is a flowchart used for explaining processing carried out by the client shown in Fig. 1 to download a content;
Fig. 4 is a flowchart used for explaining processing carried out by a content server shown in Fig. 1 to provide a client with a content;
Fig. 5 is a diagram showing a typical format of data generated at a step S26 of the flowchart shown in Fig. 4;
Fig. 6 is a flowchart used for explaining processing carried out by the client shown in Fig. 1 to play back a content;
Fig. 7 is a flowchart used for explaining details of processing carried out at a step S43 of the flowchart shown in Fig. 6 to acquire a license;
Fig. 8 is a diagram the configuration of a license;
Fig. 9 is a flowchart used for explaining processing carried out by a license server shown in Fig. 1 to issue a license;
Fig. 10 is a flowchart used for explaining details of processing carried out at a step S45 of the flowchart shown in Fig. 6 to update a license;
Fig. 11 is a flowchart used for explaining processing carried out by the license server shown in Fig. 1 to update a license;
Fig. 12 is an explanatory diagram showing an organization of keys;
Fig. 13 is an explanatory diagram showing category nodes;
Fig. 14 is a diagram concretely showing typical association of nodes with devices;
Figs. 15A and 15B are explanatory diagrams showing the configuration of an EKB (Enabling Key Block) ;
Fig. 16 is an explanatory diagram showing use of the EKB (Enabling Key Block);
Fig. 17 is an explanatory diagram showing a typical format of the EKB (Enabling Key Block);
Figs. 18A to 18C are explanatory diagrams showing the configuration of tags in an EKB (Enabling Key Block) ;
Fig. 19 is an explanatory diagram showing processing to decrypt a content by using an DNK (Device Node Key);
Fig. 20 is a diagram showing a typical EKB (Enabling Key Block) ;
Fig. 21 is an explanatory diagram showing assignment of a plurality of contents to a device;
Fig. 22 is an explanatory diagram showing license categories;
Fig. 23 is a flowchart used for explaining a ripping process carried out by a client;
Fig. 24 is an explanatory diagram showing the configuration of a watermark;
Fig. 25 is an explanatory diagram showing a typical format of a content;
Fig. 26 is a diagram showing a typical certificate of a disclosed key;
Fig. 27 is an explanatory diagram showing distribu-

tion of a content;
Fig. 28 is a flowchart used for explaining processing carried out by a client to check out a content;
Fig. 29 is an explanatory diagram showing typical tracing of an EKB (Enabling Key Block) by using tags;
Fig. 30 is a diagram showing a typical configuration of the EKB (Enabling Key Block);
Fig. 31 is an explanatory diagram showing the configuration of a mark;
Fig. 32 is a flowchart used for explaining processing carried out by a client to purchase a license;
Fig. 33 is a flowchart used for explaining processing carried out by a license server to purchase a license;
Fig. 34 is an explanatory diagram showing the configuration of a mark;
Fig. 35 is a flowchart used for explaining processing carried out by a client to catalog a certificate of the client;
Fig. 36 is a flowchart used for explaining processing carried out by a content server to catalog the certificate;
Fig. 37 is a diagram showing a typical certificate of a group;
Fig. 38 is a flowchart used for explaining processing carried out by a content server to form a group;
Fig. 39 is a diagram showing typical processing to encrypt a content key;
Fig. 40 is a flowchart used for explaining processing carried out by a client pertaining to a group;
Fig. 41 is a flowchart used for explaining processing carried out by a client to check out a license to another client;
Fig. 42 is a flowchart used for explaining processing carried out by a client to receive a license checked out by another client from the other client;
Fig. 43 is a flowchart used for explaining processing carried out by a client to play back a license checked out by another client;
Fig. 44 is a flowchart used for explaining processing carried out by a client to check in a license checked out by another client;
Fig. 45 is a flowchart representing processing carried out by a client issuing a request for the processing to check in a license to another client carrying out the license-check-in processing represented by the flowchart shown in Fig. 44;
Fig. 46 is an explanatory diagram showing generation of a MAC (Message Authentication Code);
Fig. 47 is a flowchart used for explaining processing to decrypt an ICV (Integrity Check Value) generation key;
Fig. 48 is a flowchart used for explaining other processing to decrypt the ICV generation key;
Figs. 49A and 49B are explanatory diagrams showing ICV-based management of operations to copy a license; and
Fig. 50 is an explanatory diagram showing management of licenses.

ILLUSTRATIVE EMBODIMENTS OF THE INVENTION

[0024] Fig. 1 is a block diagram showing the configuration of a content-exchanging system to which an example of the preser invention is applied. Clients 1-1 and 1-2 are connected to the Internet 2. In the following description, the clients 1-1 and 1-2 are each denoted by generic reference numeral 1 if it is not necessary to distinguish the clients 1-1 and 1-2 from each other. In this example, only two clients are shown. However, any arbitrary number of clients can be connected to the Internet 2.

[0025] In addition, a content server 3, a license server 4 and a charging server 5 are also connected to the Internet 2. The content server 3 provides contents to the client 1, and the license server 4 provides the client 1 with a license required for using a content provided by the content server 3. The charging server 5 carries out a charging process for the client 1 when the client 1 receives a license from the license server 4.

[0026] Any arbitrary number of content servers 3, license servers 4 and charging servers 5 can be connected to the Internet 2.

[0027] Fig. 2 is a block diagram showing the configuration of the client 1.

[0028] In the client 1 shown in Fig. 2, a CPU (Central Processing Unit) 21 carries out various kinds of processing in accordance with programs stored in a ROM (Read-Only Memory) 22 and programs loaded from a storage unit 28 into a RAM (Random-Access Memory) 23. A timer 20 measures the lapse of time and supplies a result of measurement to the CPU 21. The RAM 23 is also used for storing data required by the CPU 21 in the execution of the various kinds of processing.

[0029] An encryption and decryption unit 24 encrypts a content and decrypts an already encrypted content. A codec unit 25 encodes a content in accordance with an ATRAC (Adaptive Transform Acoustic Coding)-3 system and supplies the encoded content to a semiconductor memory 44 to be stored therein. The semiconductor memory 44 is connected to a drive 30 by an input/output interface 32. In addition, the codec unit 25 decodes encoded data read out from the semiconductor memory 44 through the drive 30.

[0030] An example of the semiconductor memory 44 is a memory stick (a trademark).

[0031] The CPU 21, the ROM 22, the RAM 23, the encryption and decryption unit 24, and the codec unit 25 are connected to each other by a bus 31. The bus 31 is also connected to the input/output interface 32.

[0032] The input/output interface 32 is connected to an input unit 26, an output unit 27, the storage unit 28 and a communication unit 29. The input unit 26 includes a keyboard and a mouse. The output unit 27 includes a speaker and a display unit such as a CRT and an LCD.

The communication unit 29 includes a modem and a terminal adaptor. The communication unit 29 carries out communications through the Internet 2. To be more specific, the communication unit 29 exchanges analog and digital signals with other clients.

[0033] If necessary, the input/output interface 32 is also connected to the drive 30, on which a proper storage medium such as a magnetic disk 41, an optical disk 42, a magneto-optical disk 43 or the semiconductor memory 44 is mounted. If necessary, a computer program can thus be read out from the storage medium and installed in the storage unit 28.

[0034] The configurations of the content server 3, the license server 4 and the charging server 5 are not shown in figures. However, the content server 3, the license server 4 and the charging server 5 are each a computer having a configuration basically identical with the client 1 shown in Fig. 2. For this reason, some of the reference numerals shown in the configuration of Fig. 2 are used for denoting identical components employed in the content server 3, the license server 4 and the charging server 5 in the following description.

[0035] By referring to a flowchart shown in Fig. 3, the following description explains processing to provide a content from the content server 3 to the client 1.

[0036] As shown in the figure, the flowchart begins with a step S1 at which the user enters a command to make an access to the content server 3 by operating the input unit 26. In accordance with the command, the CPU 21 controls the communication unit 29 to make an access to the content server 3 through the Internet 2. Then, at the next step S2, when the user specifies a desired content by operating the input unit 26, the CPU 21 accepts the specification. The communication unit 29 informs the content server 3 of the specified content through the Internet 2. Notified of the specified content, the content server 3 transmits encoded data of the content in a process represented by a flowchart shown in Fig. 4 as will be described later. Subsequently, at the next step S3, the CPU 21 receives the content through the communication unit 29. Then, at the next step S4, the encoded data of the content is stored in a hard disk of the storage unit 28.

[0037] With reference to the flowchart shown in Fig. 4, the following description explains the processing carried out by the content server 3 to transmit a content requested by the client 1 at the step S2. It should be noted that, since the content server 3 has a configuration comprising components identical with those employed in the client 1, the same reference numerals as those shown in Fig. 2 are used for denoting identical components in the following description.

[0038] As shown in Fig. 4, the flowchart begins with a step S21 at which the CPU 21 employed in the content server 3 is waiting for an access to arrive from the Internet 2 by way of the communication unit 29. As the access is determined to have arrived, the flow of the processing goes on to a step S22 at which information transmitted by the client 1 to specify a content is acquired. The information specifying a content is transmitted by the client 1 at the step S2 of the flowchart shown in Fig. 3.

[0039] Then, at the next step S23, the CPU 21 employed in the content server 3 reads out a content specified by the information acquired at the step S22 from the storage unit 28. The content read out from the storage unit 28 is selected among contents stored in the storage unit 28. Subsequently, at the next step S24, the CPU 21 supplies the content read out from the storage unit 28 to the encryption and decryption unit 24, which then encrypts the content by using a content key Kc.

[0040] Since the contents stored in the storage unit 28 have been encoded by the codec unit 25 in accordance with the ATRAC3 system, the encryption and decryption unit 24 encrypts an encoded content received from the CPU 21.

[0041] It should be noted that the storage unit 28 can also be used for storing contents, which have been encrypted in advance. In this case, the processing carried out at the step S24 can be eliminated.

[0042] Then, at the next step S25, the CPU 21 employed in the content server 3 adds keys and a license ID to a header of a format in which the encrypted content is to be transmitted. Required for decrypting the encrypted data, the keys are an EKB and a key K_{EKBC} (Kc), which will be described later by referring to Fig. 5. The license ID is used for identifying a license necessary for using the content. Subsequently, at the next step S26, the CPU 21 employed in the content server 3 transmits formatted data to the client 1 through the communication unit 29 and the Internet 2. The formatted data is a result of formatting the content encrypted at the step S24 and the header including the key and the license ID, which are added to the header at the step S25.

[0043] Fig. 5 is a diagram showing the format of a content received by the client 1 from the content server 3. As shown in the figure, the format comprises a header and a data portion.

[0044] The header comprises content information, DRM (Digital Right Management) information, a license ID, an EKB (Enabling Key Block) and a key K_{EKBC} (Kc), which is a content key Kc encrypted by a key K_{EKBC} generated from the EKB. It should be noted that the EKB will be explained later by referring to Fig. 15.

[0045] The content information includes a content ID CID and information on a codec system. The content ID CID is identification for identifying the formatted content in the data portion of the format.

[0046] The DRM information comprises the content's usage rules and status. The DRM information also includes a URL (Uniform Resource Locator). The usage rules and status typically include the number of times the content has been played back and the number of times the content has been copied.

[0047] The URL is an address to which an access is made in order to acquire a license prescribed by the li-

cense ID. To put it concretely, the URL is the address of the license server 4 for providing a required license in the case of the content-exchanging system shown in Fig. 1. The license ID is identification for identifying a license required for using the content recorded as the data portion of the format.

[0048] The data portion of the format comprises any arbitrary number of encrypted blocks. Each of the encrypted blocks comprises an initial vector IV, a seed and encrypted data Ek'c (data), which is a result of encrypting the content by using a key K'c.

[0049] As shown in the following formula, the key K'c is a result of processing carried out by applying the content key Kc and the seed to a hash function. The seed is a value set at random.

$$K'c = Hash\ (Kc, Seed)$$

[0050] The initial vector IV and the seed vary in dependence on the encrypted block.

[0051] A content is encrypted in 8-byte units. The 8 bytes at a current stage are encrypted by using a result of encryption of 8 bytes at the preceding stage in a CBC (Cypher Block Chaining) mode.

[0052] In the CBC mode, when the first 8 bytes of a content are encrypted, no encryption of 8 bytes is carried out at the preceding stage so that a result of preceding-stage encryption is not available. Thus, the first 8 bytes of a content are encrypted by using the initial vector IV as an initial value.

[0053] Therefore, even if an encrypted block of a content encrypted in the CBC mode can be decoded, the decoding result of the block may not necessarily make another encrypted block easy to decode.

[0054] It should be noted that the encryption process will be explained later in detail by referring to Fig. 46. The encryption system, however, is not limited thereto.

[0055] As described above, the client 1 is allowed to freely acquire a content from the content server 3 for free of charge. Thus, a large number of contents themselves can be distributed.

[0056] When using an acquired content, however, the client 1 needs to has a license. The following description explains processing carried out by the client 1 to play back a content with reference to a flowchart shown in Fig. 6.

[0057] As shown in the figure, the flowchart begins with a step S41 at which the user operates the input unit 26 to specify a desired content and the CPU 21 employed in the client 1 acquires the identification of the content (CID). The identification comprises the title of the content, a number assigned to the content and the like. It should be noted that a number is assigned to each stored content.

[0058] When a content is desired, the CPU 21 reads out a license ID for the content. To be more specific, the license ID identifies a license required in utilization of the content. As shown in Fig. 5, the license ID is included in the header of the encrypted content.

[0059] Then, at the next step S42, the CPU 21 forms a judgment as to whether or not a license indicated by the license ID obtained at the step S41 has been acquired by the client 1 and stored in the storage unit 28. If the license indicated by the license ID has not been acquired by the client 1, the flow of the processing goes on to a step S43 at which the CPU 21 carries out processing to acquire the license. Details of the processing to acquire the license will be explained later by referring to a flowchart shown in Fig. 7.

[0060] If the outcome of the judgment formed at the step S42 indicates that the license identified by the license ID has been acquired by the client 1 and stored in the storage unit 28 or if the license can be obtained as a result of the processing carried out at the step S43 to acquire the license, the flow of the processing goes on to a step S44 at which the CPU 21 forms a judgment as to whether or not the acquired license is still in its term of validity. It is possible to determine whether or not the acquired license is still in its term of validity by comparing a term of validity prescribed in the license (which will be described in Fig. 8) with the present date and time measured by the timer 20. If the term of validity of the license has already expired, the flow of the processing goes on to a step S45 at which the CPU 21 carries out processing to update the license. Details of the processing to update a license will be explained later by referring to a flowchart shown in Fig. 8.

[0061] If the outcome of the judgment formed at the step S44 indicates that the acquired license is still in its term of validity or if the license can be updated at the step S45, the flow of the processing goes on to a step S46 at which the CPU 21 reads out the encrypted content from the storage unit 28 and stores the content into the RAM 23. Then, at the next step S47, the CPU 21 supplies the data of the encrypted content stored in the RAM 23 to the encryption and decryption unit 24 in encrypted-block units included in the data portion shown in Fig. 5. Then the encryption and decryption unit 24 decrypts the encrypted content by using a content key Kc.

[0062] A typical method of acquiring the content key Kc will be described later by referring to Fig. 15. The key K_{EKBC} included in the EKB shown in Fig. 5 can be obtained by using a device node key (DNK) shown in Fig. 8. The content key Kc is then obtained from the data K_{EKBC} (Kc) by using the key K_{EKBC}.

[0063] Subsequently, at the next step S48, the CPU 21 supplies the content decrypted by the encryption and decryption unit 24 to the codec unit 25, which then decodes the content. Subsequently, the CPU 21 supplies data decoded by the codec unit 25 to the output unit 27 by way of the input/output interface 32. The data is subjected to D/A conversion before being output to a speaker.

[0064] With reference to the flowchart shown in Fig. 7, the following description explains the processing car-

ried out at the step S43 of the flowchart shown in Fig. 6 to acquire a license.

[0065] The client 1 acquires beforehand service data cataloged in advance in the license server 4. The service data includes a leaf ID, a DNK (Device Node Key), a pair of secret and disclosed keys pertaining to the client 1, a disclosed key of the license server 4 and certificates for the disclosed keys.

[0066] A leaf ID is an ID assigned uniquely to each client 1. A DNK (Device Node Key) is a key required for decrypting an encrypted content key Kc included in the EKB (Enabling Key Block) for the license as will be described later by referring to Fig. 12.

[0067] As shown in Fig. 7, the flowchart begins with a step S61 at which the CPU 21 acquires a URL for the ID of the license being processed from the header shown in Fig. 5. As described earlier, this URL is an address to which an access is to be made in order to acquire the license identified by the license ID included in the header. Then, at the next step S62, the CPU 21 makes an access to the URL acquired at the step S61. To put it concretely, an access to the license server 4 is made through the communication unit 29 and the Internet 2. At that time, the license server 4 requests the client 1 to transmit a user ID, a password and license-specifying information specifying a license to be purchased. This license is required in utilization of the content. This request is made at a step S102 of the flowchart shown in Fig. 9 as will be described later. The CPU 21 displays this request on the display device of the output unit 27. In response to this request, the user of the client 1 operates the input unit 26 to enter a user ID, a password and license-specifying information. It should be noted that the user obtained the user ID and the password before by making an access to the license server 4 through the Internet 2.

[0068] Subsequently, at the next steps S63 and S64, the CPU 21 receives the license-specifying information, the user ID and the password, which have been entered from the input unit 26 by the user. Then, at the next step S65, the CPU 21 controls the communication unit 29 to transmit a license request including the user ID, the password, the license-specifying information and a leaf ID to the license server 4 by way of the Internet 2. The leaf ID is information included in the service data to be described later.

[0069] As will be described later, at a step S109 of the flowchart shown in Fig. 9, the license server 4 transmits a license based on the user ID, the password and the license-specifying information. As an alternative, the license server 4 carries out error processing at a step S112 of the flowchart shown in Fig. 9 instead of transmitting a license in case a condition is not met.

[0070] The processing then goes on to a step S66 to form a judgment as to whether or not the license has been received from the license server 4. If the license has been received, the flow of the processing goes on to a step S67 at which the CPU 21 supplies the license to the storage unit 28 to be stored therein.

[0071] If the outcome of the judgment formed at the step S66 indicates that the license was not received, on the other hand, the flow of the processing goes on to a step S68 at which the CPU 21 carries out error processing. To put it concretely, the CPU 21 inhibits the processing to play back the content since the CPU 21 has failed to obtain the license for using the content.

[0072] As described above, the client 1 is capable of using a content by obtaining a license indicated by a license ID included in the content.

[0073] It should be noted that the processing to acquire a license as represented by the flowchart shown in Fig. 7 can also be carried out in advance before the user acquires a content.

[0074] As shown in Fig. 8, the license supplied to the client 1 includes usage conditions and a leaf ID.

[0075] The usage conditions are information including a deadline before the content can be downloaded as allowed by the license, an upper limit of the number of times the content can be copied as allowed by the license or the maximum number of allowed copy operations, the number of checkouts, the maximum number of checkouts, a right to record the content on a CD-R, the number of times the content can be copied to a PD (Portable Device), a right to allow the license of using the content to be changed to license ownership status (or purchased-license status) and a duty of making a usage log.

[0076] By referring to the flowchart shown in Fig. 9, the following description explains processing carried out by the license server 4 to transmit a license to the client 1 in response to the processing carried out by the client 1 to acquire the license as represented by the flowchart shown in Fig. 7. It should be noted that, also in this case, since the license server 4 has a configuration comprising components identical with those employed in the client 1, the same reference numerals as those shown in Fig. 2 are used for denoting identical components in the following description.

[0077] As shown in the figure, the flowchart begins with a step S101 at which the CPU 21 employed in the license server 4 waits for an access to be made by the client 1. As the client 1 makes an access to the license server 4, the flow of the processing goes on to a step S102 at which the license server 4 requests the client 1 making the access to transmit, license-specifying information (a license ID), a user ID and a password. Then, the CPU 21 employed in the license server 4 receives the user ID, the password, the leaf ID and the license ID from the client 1 through the communication unit 29 at the step S65 of the flowchart shown in Fig. 7, and carries out processing to accept them.

[0078] Then, at the next step S103, the CPU 21 employed in the license server 4 makes an access to the charging server 5 through the communication unit 29 in order to make a request for processing to examine the user identified by the user ID and the password. When

receiving the request for such an examination from the license server 4 through the Internet 2, the charging server 5 checks the past payment history of the user identified by the user ID and the password in order to determine whether or not there is a record indicating that the user did not pay a price for the license in the past. If there is no record indicating that the user did not pay a price for the license in the past, the charging server 5 transmits an examination result indicating that granting of a license is approved. If there is a record indicating that the user did not pay a price for the license in the past or another bad record, on the other hand, the charging server 5 transmits an examination result indicating that granting of a license is not approved.

[0079] Subsequently, at the next step S104, the CPU 21 employed in the license server 4 forms a judgment as to whether the examination result received from the charging server 5 indicates that granting of a license is approved or disapproved. If granting of a license is approved, the flow of the processing goes on to a step S105 at which the CPU 21 selects a license specified by the license-specifying information received in the processing carried out at the step S102 among licenses stored in the storage unit 28, and reads out the selected license from the storage unit 28. Each license stored in the storage unit 28 includes information including the license ID, a version, a creation date and time and a term of validity. Then, at the next step S106, the CPU 21 adds the received leaf ID to the license. Furthermore, at the next step S107, the CPU 21 selects a usage condition associated with the license selected at the step S105. If a usage condition was found at the step S102 to have been specified by the user, if necessary, the usage condition specified by the user is added to a usage condition prepared in advance. The CPU 21 then adds the selected usage condition to the license.

[0080] Then, at the next step S108, the CPU 21 puts a signature on the license by using a secret key of the license server 4 to produce a license with a configuration like one shown in Fig. 8.

[0081] Subsequently, at the next step S109, the CPU 21 employed in the license server 4 transmits the license with a configuration like the one shown in Fig. 8 from the communication unit 29 to the client 1 by way of the Internet 2.

[0082] Then, at the next step S110, the CPU 21 employed in the license server 4 stores the license transmitted in the processing carried out at the step S109 in the storage unit 28 by associating the license with the user ID and the password, which were acquired in the processing carried out at the step S102. As described above, the license includes a usage condition and a leaf ID. Subsequently, at the next step S111, the CPU 21 carries out a charging process. To put it concretely, the CPU 21 requests the charging server 5 through the communication unit 29 to carry out a charging process for a user identified by the user ID and the password. The charging server 5 carries out the charging process based on the request for the user. The processing carried out by the license server 4 is then finished. If the user does not pay an amount of money determined by the charging process, in the future, the user will not be granted a license even if granting of a license is requested as described above.

[0083] That is to say, in the case of such a user, the charging server 5 does not approve granting of a license as a result of examination of whether or not to grant a license to the user. That is to say, the flow of the processing goes on from the step S104 to a step S112 at which the CPU 21 employed in the license server 4 carries out an error-handling process. To put it concretely, the CPU 21 controls the communication unit 29 to output a message informing the client 1 making the access that a license cannot be granted. The processing carried out by the license server 4 is then finished.

[0084] In this case, the client 1 cannot use the content or is not capable of decrypting the encrypted data of the content because the client 1 has failed to obtain a license as described above.

[0085] Fig. 10 is a flowchart used for explaining details of the processing carried out at the step S45 of the flowchart shown in Fig. 6 to update a license. Processing carried out at steps S131 to S135 is basically the same as the processing carried out at respectively the steps S61 to S65 of the flowchart shown in Fig. 7. At the step S133, however, the CPU 21 acquires the license ID of a license to be updated instead of the license ID of a purchased license. Then, at the step S135, the CPU 21 transmits a user ID and a password along with the license ID of the license to be updated to the license server 4.

[0086] In response to what is transmitted in the processing carried out at the step S135, the license server 4 presents usage conditions at a step S153 of a flowchart shown in Fig. 11 as will be described later. Then, at the next step S136, the CPU 21 employed in the client 1 receives the presented usage conditions from the license server and displays them on the output unit 27. The user operates the input unit 26 to select one of the displayed usage conditions and/or newly add a predetermined usage condition. Subsequently, at the next step S137, the CPU 21 transmits an application to purchase a usage condition selected as described above or a condition to update a license to the license server 4. In response to the application, the license server 4 transmits an eventual usage condition at a step S154 of the flowchart shown in Fig. 11 as will be described later. Then, at the next step S138, the CPU 21 employed in the client 1 receives the eventual usage condition from the license server 4. Subsequently, at the next step S139, the received eventual usage condition is used as an update of the license's usage condition already stored in the storage unit 28.

[0087] Fig. 11 is a flowchart used for explaining the processing carried out by the license server 4 to update a license in conjunction with the processing carried out

by the client 1 to request an operation to update the license.

[0088] As shown in the figure, the flowchart begins with a step S151 at which the license server 4 is accessed by the client 1. Then, at the next step S152, the CPU 21 employed in the license server 4 receives a request to update a license from the client 1 along with the license-specifying information received by the client 1 at the step S135.

[0089] Subsequently, at the next step S153, the CPU 21 reads out a usage condition for a license to be updated in accordance with the request to update the license, or reads out a condition for updating the license from the storage unit 28. The CPU 21 then transmits the condition to the client 1.

[0090] In response to the transmitted condition, assume that the user of the client 1 enters an application to purchase the usage condition in processing carried out at the step S137 of the flowchart shown in Fig. 10. In this case, at the next step S154, the CPU 21 employed in the license server 4 generates data for the applied usage condition and transmits the data to the client 1. As described above, the client 1 uses the received usage condition as an update of the already cataloged usage condition of the license. As described above, the usage condition was updated in the processing carried out at the step S139.

[0091] In the present embodiment, keys of devices and keys of licenses are managed on the basis of the principle of a broadcast-encryption system as shown in Fig. 12. The keys are organized into a hierarchical tree structure. Leaves, which are at the bottom hierarchical layer, each corresponds to keys of each device. In the typical structure shown in Fig. 12, 16 keys 0 to 15 corresponding to 16 devices or 16 licenses are generated.

[0092] Each key denoted by a circular mark is placed at a node of the tree structure. A root key KR is placed at a root node on the top of the tree structure. At nodes on the second hierarchical layer, keys K0 and K1 are provided. At nodes on the third hierarchical layer, keys K00 to K11 are placed. At nodes on the fourth hierarchical layer, keys K000 to K111 are provided. The leaves at nodes on the bottom hierarchical layer or device nodes are keys K0000 to K1111.

[0093] In the hierarchical tree structure, for example, keys K0010 and K0011 are each a subordinate of key K001. In the same way, keys K000 and K001 are each a subordinate of key K00. By the same token, on the higher hierarchical layers, keys K00 and K01 are each a subordinate of key K0. Likewise, keys K0 and K1 are each a subordinate of the root key KR.

[0094] Keys required for using a content comprise a leaf at a device node of the bottom hierarchical layer and keys at nodes on higher hierarchical layers including the root key KR. The leaf and the keys on the higher hierarchical layers including the root key KR form a path. For example, keys required for using Content 3 are managed by each key of the path including keys K0011, K001, K00, K0 and KR, which form a path starting with the leaf K0011 and ending with the root key KR on the basis of the license corresponding to the leaf ID.

[0095] The content-exchanging system provided by the present embodiment adopts typically the principle shown in Fig. 12 to manage keys of devices and licenses placed at nodes laid out to form a 8 + 24 + 32-layer structure shown in Fig. 13. In the structure shown in Fig. 13, there are eight subordinate hierarchical layers below the root node. A key at each node of the eight hierarchical layers is associated with a category. Examples of the category are a category of equipment using a semiconductor memory such as a memory stick and a category of equipment for receiving digital broadcasts.

[0096] One of the category nodes is the root node of a system called a T system provided by the present embodiment for managing licenses.

[0097] To put it in detail, subordinates of the root node of the T system are nodes on 24 hierarchical layers. A key at each of the subordinate nodes is associated with a license. Thus, it is possible to prescribe 2^{24} licenses or about 16 mega or about 1.6 million licenses. Further subordinates on the lower side are 32 hierarchical layers, which allow 2^{32} users (or clients 1) or about 4 giga or about 4 billion users (or clients 1) to be prescribed. Keys placed at nodes on the 32 hierarchical layers are each a DNK (Device Node Key).

[0098] Keys for a device or keys for a license are placed along a path passing through nodes at the 64 (= 8 + 24 + 32) hierarchical layers. Thus, such a path is associated with a device or a license. To put it concretely, content keys used for encrypting a content are encrypted by keys placed at nodes passed through by a path associated with a license for the content. A key at an upper hierarchical layer is encrypted by using its direct subordinate key on a directly below hierarchical layer and put in an EKB to be described later by referring to Fig. 15. A DNK on the bottom hierarchical layer is not put in the EKB, but included in service data to be granted to the client 1 of the user. The client 1 uses a DNK included in the license to decrypt a key, which is placed on a hierarchical layer directly above the DNK and included in the EKB shown in Fig. 15. The EKB is distributed along with the data of the content. The client 1 then uses the decrypted key to decrypt a key, which is placed on a hierarchical layer directly above the decrypted key and included in the EKB. This decryption process is carried out repeatedly till the client 1 is capable of obtaining all keys placed at nodes passed through by the path associated with the license.

[0099] Fig. 14 is a diagram showing typical classification of categories each associated with a key of a hierarchical tree structure. As shown in Fig. 14, on the top of the hierarchical tree structure, a root key KR2301 is set. On an intermediate hierarchical layer under the root key KR2301, a node key 2302 is set. On the bottom hierarchical layer, a leaf key 2303 is set. Each device has a leaf key, node keys and the root key. The nodes keys

owned by the device are provided along a path, which is associated with the device and connects the leaf key to the root key.

[0100] A predetermined node on an Mth hierarchical layer from the root key is set as a category node 2304. In the example shown in Fig. 13, let M be 8. Each node on the Mth hierarchical layer is used as a node for setting a root key for a device pertaining to a specific category. That is to say, a device of a category is associated with a path starting at a node on the Mth hierarchical layer, passing through nodes on (M + 1)th and lower hierarchical layers, and ending at a leaf on the bottom hierarchical layer.

[0101] Assume that, at a node 2305 on the Mth hierarchical layer of the hierarchical tree structure shown in Fig. 14, a category of a memory stick (trademark) is set. In this case, linked nodes and leaves below this node 2305 are used as nodes and leaves provided specially for the category including a variety of devices each using the memory stick. That is to say, nodes and a leaf under the node 2305 are defined as a set of nodes and a leaf, which are related to a device defined in the category of the memory stick.

[0102] In addition, a node on a hierarchical layer several levels below the Mth hierarchical layer can be set as a subcategory node 2306. In the hierarchical tree structure shown in Fig. 14, a node on a hierarchical layer two levels below the memory-stick-category hierarchical layer, on which the node 2305 is provided, is set as a subcategory node, that is, a node for a subcategory included in the category of devices each using a memory stick. Furthermore, on a hierarchical layer under the node 2306 for a subcategory of playback-only equipment, a node 2307 for a telephone with a function to play back music is set. Such a telephone is included in the category of playback-only equipment. Moreover, on a hierarchical layer under the node 2307, a PHS node 2308 and a cellular-phone node 2309 are provided. A PHS and a cellular phone are included in the category of a telephone with a function to play back music.

[0103] In addition, categories and subcategories are provided not only for devices but also for nodes controlled by a manufacturer, a content provider and a charging institution independently or any arbitrary units, which can be processing units, control units, presentation service units or the like. These units are each referred to as an entity, which is a generic technical term. Assume that a category node is set as a root node provided specially for a game machine XYZ sold by a game-machine manufacturer. In this case, the game-machine manufacturer is capable of selling the game machine XYZ by storing node keys and a leaf key, which are provided on hierarchical layers below the root node, in the game machine XYZ. Then, encrypted contents or a variety of keys are distributed, or the keys are updated by generating an EKB. The EKB comprises of node keys and a leaf key, which are provided on hierarchical layers below the root node. In this way, it is possible to distribute data usable only by a device under the root node.

[0104] As described above, a node is used as a root node and nodes on hierarchical layers below the root node are each set as a category-related node or a subcategory-related node. In this way, an institution such as a manufacturer or a content provider, which manages a root node on a category hierarchical layer or a subcategory hierarchical layer, is capable of generating an EKB (Enabling Key Block) with a key at the root node used as the root key thereof by itself and distributing the EKB to devices pertaining to hierarchical layers below the root node. Thus, it is possible to update the keys of the EKB without no effects at all on devices pertaining to a category node not serving as a subordinate to the root node.

[0105] Assume that, in the tree structure shown in Fig. 12, four devices 0, 1, 2 and 3 pertaining to a group share common keys K00, K0 and KR as node keys. By using this node-key-sharing configuration, a common content key can be provided only to devices 0, 1, 2 and 3. For example, node key K00 itself, which is shared by devices 0, 1, 2 and 3, is set as the common content key. In this way, it is possible to set a content key common to devices 0, 1, 2 and 3 without transmitting a new key. As an alternative, a new content key Kcon is encrypted by using the node key K00 to result in an encrypted value Enc(K00, Kcon), which is then distributed to devices 0, 1, 2 and 3 by way of a network or by storing the value Enc(K00, Kcon) in a storage medium distributed to devices 0, 1, 2 and 3. In this way, only devices 0, 1, 2 and 3 are capable of decrypting the value Enc(K00, Kcon) by using the common node key K00 shared by devices 0, 1, 2 and 3 to produce the content key Kcon. It should be noted that notation Enc(Ka, Kb) denotes data obtained as a result of encryption of a key Kb by using a key Ka.

[0106] In addition, assume that it is discovered at a point time t that keys K0011, K001, K00, K0 and KR owned by device 3 have been analyzed and identified by a hacker. In this case, in order to protect data exchanged thereafter in a system (or a group of devices 0, 1, 2 and 3), device 3 needs to be detached from the system. In addition, node keys K001, K00, K0 and KR need to be updated to respectively new keys K(t)001, K(t)00, K(t)0 and K(t)R to be transmitted to devices 0, 1 and 2. It should be noted that notation K(t)aaa is an updated key of generation of time point t and is obtained by updating a key Kaaa.

[0107] Processing to distribute an updated key is explained below. For example, keys are updated as follows. In the case of the processing to update keys for devices 0, 1 and 2 as described above, a table is supplied to devices 0, 1 and 2 by way of a network or by storing the table in a storage unit and supplying the storage unit to devices 0, 1 and 2. The table is block data called an EKB (Enabling Key Block) shown in Fig. 15A. It should be noted that the EKB (Enabling Key Block) comprises encrypted keys for distributing newly updat-

ed keys to devices associated with leaves at the nodes on the bottom hierarchical layer of a tree structure like the one shown in Fig. 12. The EKB (Enabling Key Block) is also called a KRB (Key Renewal Block).

[0108] The EKB (Enabling Key Block) shown in Fig. 15A is structured as block data that can be updated by only a device requiring node keys thereof to be updated. The typical EKB shown in Fig. 15A is block data created with an objective of distributing updated node keys of generation of time point t to devices 0, 1 and 2 of the tree structure shown in Fig. 12. As is obvious from Fig. 12, devices 0 and 1 each require K(t)00, K(t)0 and K(t)R as updated node keys. On the other hand, device 2 requires K(t)001, K(t)00, K(t)0 and K(t)R as updated node keys.

[0109] As shown in Fig. 15A, the EKB includes a plurality of encrypted keys. An encrypted key of the fifth column of the table from the top shown in Fig. 15A is Enc(K0010, K(t)001), which is an updated node key K(t)001 encrypted by using a leaf key K0010 owned by the device 2. The device 2 is thus capable of obtaining the updated node key K(t)001 by decryption of the encrypted key Enc(K0010, K(t)001) by using the leaf key K0010 owned by device 2 itself. In addition, by using the updated node key K(t)001 obtained as a result of the decryption, it is possible to decrypt an encrypted key Enc(K(t)001, K(t)00) of the fourth column of the table shown in Fig. 15A to result in an updated node key K(t)00.

[0110] Thereafter, in such a sequential decryption process, an encrypted key Enc(K(t)00, K(t)0) of the second column of the table shown in Fig. 15A is decrypted to result in the updated node key K(t)0, which is then used for decrypting an encrypted key Enc(K(t)0, K(t)R) of the first column to result in the updated node key K(t)R.

[0111] On the other hand, a node key K000 is not a key to be updated. Updated node keys required by devices 0 and 1 associated with nodes 0 and 1 respectively are K(t)00, K(t)0 and K(t)R. Devices 0 and 1 associated with nodes 0 and 1 respectively each use device keys K0000 and K0001 to decrypt an encrypted key Enc (K000, K(t)00) of the third column of the table shown in Fig. 15A to obtain an updated node key K(t)00. Thereafter, in the sequential decryption process, an encrypted key Enc(K(t)00, K(t)0) of the second column of the table shown in Fig. 15A is decrypted to result in the updated node key K(t)0, which is then used for decrypting an encrypted key Enc(K(t)0, K(t)R) of the first column to result in the updated node key K(t)R. In this way, devices 0, 1 and 2 are each capable of obtaining the updated node key K(t)R.

[0112] It should be noted that indexes shown in Fig. 15A are absolute addresses of node keys and leaf keys. The node keys and leaf keys are each used as a decryption key for decrypting an encrypted key on the right side of the figure.

[0113] When it is not necessary to update a node key K(t)0 on an upper hierarchical layer of the tree structure shown in Fig. 12 and the root key K(t)R, while it is necessary to carry out processing to update only a node key K00, an updated node key K(t)00 can be distributed to devices 0, 1 and 2 by using an EKB (Enabling Key Block) shown in Fig. 15B.

[0114] The EKB shown in Fig. 15B can be used for distributing typically new content keys common to devices pertaining to a specific group. Assume that devices 0, 1, 2 and 3 pertaining to a group enclosed by a dotted line in Fig. 12 each use a recording medium and require a new common content key K(t)con. In this case, data Enc(K(t)00, K(t)con) is distributed to devices 0, 1, 2 and 3 along with the EKB shown in Fig. 15B. The data Enc(K(t)00, K(t)con) is a result of encryption of the newly updated content key K(t)con common to devices 0, 1, 2 and 3. The newly updated content key K(t)con is encrypted by using K(t)00 which is a result of encryption of a node key K00 common to devices 0, 1, 2 and 3. By this, the encrypted data Enc(K(t)00, K(t)con) can be distributed so that equipment of other groups such as device 4 cannot decrypt the encrypted data.

[0115] That is to say, devices 0, 1 and 2 are each capable of decrypting encrypted data by using the key K(t)00 obtained as a result of EKB processing in order to obtain the content key K(t)con of generation of time point t.

[0116] Fig. 16 is a diagram showing typical processing carried out by device 0, which has received the data Enc(K(t)00, K(t)con) and the EKB shown in Fig. 15B through a recording medium, to obtain the content key K(t)con of generation of time point t. As described earlier, the data Enc(K(t)00, K(t)con) is a result of encryption of the newly updated content key K(t)con common to devices 0, 1, 2 and 3. That is to say, in this typical processing, encrypted message data distributed by using the EKB is the content key K(t)con.

[0117] As shown in Fig. 16, in the same EKB processing as that described above, device 0 generates a node key K(t)00 by using an EKB of generation of time point t and a node key K000 stored in advance in the recording medium by itself. The EKB has been stored in the recording medium. Then, device 0 uses the updated node key K(t)00 obtained as a result of decryption to decrypt the updated content key K(t)con. Device 0 then encrypts it by using a leaf key K0000 owned only by device 0 for later use.

[0118] Fig. 17 is a diagram showing a typical format of an EKB (Enabling Key Block). A version 601 is an identifier indicating a version of the EKB (Enabling Key Block). It should be noted that the version 601 has functions of determining the most recent EKB and of indicating the relation with contents. A depth is the number of hierarchical layers in a hierarchical-tree structure for a device serving as a destination of the EKB (Enabling Key Block). A data pointer 603 is a pointer pointing to a position in a data portion 606 of the EKB (Enabling Key Block). A tag pointer 604 is a pointer pointing to the po-

sition of a tag 607 and a signature pointer 605 is a pointer pointing to the position of a signature 608.

[0119] The data portion 606 is typically encrypted updated node keys, that is, encrypted keys obtained as a result of encryption of updated node keys as shown in Fig. 16.

[0120] The tag 607 is a tag showing a positional relation between encrypted node keys and encrypted leaf keys. The encrypted node keys and the encrypted leaf keys are included in the data portion 606. A rule of providing a tag is explained by referring to Fig. 18.

[0121] Fig. 18 is a diagram showing an example of transmitting an EKB (Enabling Key Block) explained earlier by referring to Fig. 15A. Data in this case is shown in Fig. 18B. The address of a top node included in an encrypted key at that time is referred to as a top-node address. In this example, since an updated root key K(t)R is included, the top-node address is KR. In this case, for example, data Enc(K(t)0, K(t)R) of the first column corresponds to a position P0 in a hierarchical tree structure shown in Fig. 18A. Data Enc(K(t)00, K(t)0) of the second column corresponds to a position P00 on the lower left side of the position P0 in the hierarchical tree structure. If there is data on a hierarchical layer below a predetermined position of the hierarchical tree structure, the tag is set at 0. If there is no data on a hierarchical layer below a predetermined position of the hierarchical tree structure, on the other hand, the tag is set at 1. Tags are set in the following format: {Left (L) tag, Right (R) tag} explained below. At the position P00 on the lower left side of the position P0 corresponding to the data Enc(K(t)0, K(t)R) of the first column shown in Fig. 18B, there is data. Thus, the L tag is set at 0. At a position on the lower right side of the position P0, on the other hand, there is no data. In this case, the R tag is set at 1. In this way, for each data, tags are set. Fig. 18C is a diagram showing a configuration including a typical array of pieces of data and an array of tags.

[0122] A tag is set to indicate which position in the tree structure the corresponding data Enc(Kxxx, Kyyy) is located at. The key data Enc(Kxxx, Kyyy) and so on stored in the data portion 606 is no more than an array of keys, which have been encrypted in a simple way. With the tag described above, however, it is possible to identify the position of an encrypted key, which is stored as data, in the tree structure. Without the tag described above, it is also possible to construct data by using node indexes associated with pieces of encrypted data as is the case with the configuration explained earlier by referring to Fig. 15. An example of the data construction is given as follows:

0: Enc(K(t)0, K(t)R)
00: Enc(K(t)00, K(t)0)
000: Enc(K(t)000, K(t)00)

[0123] In a configuration using such indexes, however, redundant data is resulted in and the amount of data thus increases. As a result, such a configuration is not desirable for distribution through a network or for other purposes. By using tags as index data showing the positions of keys, however, the positions of keys can be recognized by using only a small amount of data.

[0124] The EKB format is further explained by referring back to Fig. 17. The signature 608 is an electronic signature put by an institution issuing the EKB (Enabling Key Block). Examples of such an institution are a key management center (the license server 4), a content provider (the content server 3) and a charging institution (the charging server 5). A device receiving the EKB confirms that the EKB is a valid EKB issued by an authorized EKB issuer by signature authentication.

[0125] It is possible to summarize processing to utilize a content supplied by the content server 3 on the basis of a license issued by the license server 4 as described above into what is shown in Fig. 19.

[0126] As shown in the figure, when the content server 3 provides a content to the client 1, the license server 4 issues a license to the client 1. The content is Enc(Kc, Content), which is a notation indicating that the content has been encrypted by a content key Kc. The content key Kc is encrypted by using a root key KR to produce Enc(KR, Kc). The root key KR is obtained from the EKB and corresponds to the key K_{EKBC} shown in Fig. 5. The content key Enc(KR, Kc) and the EKB are then added to the encrypted content. The content key Enc(KR, Kc), the EKB and the encrypted content are finally supplied to the client 1.

[0127] As shown in Fig. 20, the EKB in the example shown in Fig. 19 typically includes Enc (DNK, KR), which is a notation indicating that the root key KR has been encrypted by a DNK. Thus, by using a DNK included in service data, the client 1 is capable of obtaining the root key KR from the EKB. Then, it is possible to obtain the content key KC by decryption of Enc(KR, Kc) using the root key KR. Finally, it is possible to obtain the content by decryption of Enc(Kc, Content) using the content key Kc.

[0128] By assigning a DNK to each client 1 in this way, it is also possible to individually revoke a client 1 in accordance with the principles shown in Figs. 12 and 15.

[0129] In addition, by including an additional license leaf ID as a part of data in the distribution, service data is associated with a license so that it is possible to avoid an illegal copy operation in the client 1.

[0130] Furthermore, by distributing a secret key and a certificate for each client as a service data, it is possible to create a content for which the secret key and a certificate for use by the client 1 are used to prevent the end user from carrying out an illegal operation to copy the content.

[0131] The use of the secret key and a certificate will be described later by referring to a flowchart shown in Fig. 28.

[0132] As described earlier by referring to Fig. 13, in accordance with the present embodiment, a T system for managing licenses at a category node is associated with a category of devices each used for using contents.

Thus, a plurality of DNKs can be owned by the same device. As a result, contents pertaining to different categories can be managed by using one device.

[0133] Fig. 21 is an explanatory diagram showing assignment of a plurality of contents to one device. To be more specific, a license for using content 1, to which DNK 1 is assigned, is recorded in a device D1 on the basis of the T system. By the same token, content 2 to which DNK 2 is assigned can be recorded in the device D1 by transferring content 2 from a CD to a memory stick. In this way, the device D1 is capable of simultaneously handling two contents, namely, contents 1 and 2, which are distributed by different systems, namely, the T system and a device management system. This feature cannot be implemented in a case where only one DNK is assigned to a device. An example of such a case is a case in which an already assigned DNK is deleted when a new DNK is assigned.

[0134] In addition, for example, license categories 1 and 2 shown in Fig. 22 are assigned to each triangle of the 32 lower-side hierarchical layers shown in Fig. 13. By such assignment, a category is classified into subcategories for managing smaller groups such as genres of the content, levels of the content, retail stores of the content and distribution services of the content.

[0135] In the typical assignment shown in Fig. 22, for instance, license categories 1 and 2 pertain to a jazz genre and a rock genre respectively. License category 1 is associated with contents 1 and 2, each of which has a license ID of 1 and is distributed to users 1 and 3. License category 2 includes contents 3, 4 and 5, each of which has a license ID of 2 and is distributed to users 1 and 3.

[0136] As described above, in accordance with the present embodiment, independent key management can be executed for each category.

[0137] In addition, instead of having a DNK embedded in equipment and/or media, a DNK can also be downloaded to each equipment and/or each media in catalog processing carried out by the license server 4 so as to implement a system allowing a user to purchase the key.

[0138] It is desirable to provide a content that can be used in all applications by adopting any technique of using the content after creation of the content without regard to what technique is adopted. For example, it is desirable to provide a content that can be used in domains with different content distribution services or different usage conditions. In order to provide such a content, according to the present embodiment, the license server 4 functioning as an authenticating station distributes secret keys and certificates of disclosed keys for the secret keys to users (clients 1) as described above. Then, the users each use a secret key to create a signature to be put on a content in order to assure the integrity of the content and, thus, to prevent the content from being falsified.

[0139] Typical processing of the case described above is explained by referring to a flowchart shown in Fig. 23. To be more specific, the processing is a ripping process carried out by the user to record data played back from a CD in the storage unit 28.

[0140] As shown in the figure, the flowchart begins with a step S171 at which the CPU 21 employed in the client 1 receives inputted recorded data played back from a CD from the communication unit 29. Then, at the next step S172, the CPU 21 forms a judgment as to whether or not the recorded data input at the step S171 includes a watermark embedded in the data of the content. The watermark comprises 3-bit CCI (Copy Control Information) and a 1-bit trigger. If a watermark is detected, the flow of the processing goes on to a step S173 at which the CPU 21 carries out a process to extract the watermark. If no watermark is detected, on the other hand, the watermark-extracting process is skipped.

[0141] Then, at the next step S174, the CPU 21 creates data of a header to be recorded for the content. The data of the header comprises a content ID, a license ID, a URL representing an access target for acquiring a license and a watermark.

[0142] Subsequently, at the next step S175, by using the secret key of the CPU 21 itself, the CPU 21 creates a digital signature based on the data of the header created in the processing carried out at the step S174. The secret key has been obtained from the license server 4 at the step S67 of the flowchart shown in Fig. 7.

[0143] Then, at the next step S176, the CPU 21 controls the encryption and decryption unit 24 to encrypt the content by using a content key. The content key has been acquired at the same time when acquiring the content (See Fig. 5 or 9).

[0144] Subsequently, at the next step S177, CPU 21 records the data onto a magneto-optical disk 43 in a file format. Typically, the magneto-optical disk 43 is a mini disc.

[0145] It should be noted that, in the case of a mini disk used as the recording medium, the CPU 21 supplies the content to the codec unit 25 at the step S176. The codec unit 25 encodes the content in accordance with typically the ATRAC3 system. The encoded content is further encrypted by the encryption and decryption unit 24.

[0146] Fig. 24 is a diagram showing a model of a content recorded on the recording medium. A watermark WM extracted from the encrypted content (E(At3)) is recorded in a header outside the content.

[0147] Fig. 25 is a diagram showing a more detailed configuration of the file format in which a content is recorded onto the recording medium. As is obvious from the typical configuration, a header including a content ID (CID), a license ID (LID), a URL and a watermark (WM) is recorded. In addition, an EKB, data Enc(KR, Kc), a certificate (Cert), a digital header (Sig(Header)), data Enc(Kc, Content), meta data and a mark are recorded. The data Enc(Kr, Kc) is a result of encryption of a content key Kc by using a root key KR, whereas the

data Enc(Kc, Content) is a result of encryption of the content by using the content key Kc. The digital header Sig(Header) has been generated on the basis of the header.

[0148] The watermark is embedded in the content. As shown in Figs. 24 and 25, in addition to the inside of the content, the watermark is also placed in the header so that information embedded in the content as the watermark can be detected fast and with ease. Thus, it is possible to quickly form a judgment as to whether or not the content can be copied.

[0149] It should noted that the meta data typically represents a jacket, pictures, a libretto and other information. The mark will be described later by referring to Fig. 31.

[0150] Fig. 26 is a diagram showing a typical disclosed-key certificate used as the certificate of a disclosed key. Normally, a disclosed-key certificate is a certificate issued by a CA (Certificate Authority) in a disclosed-key encryption system. A disclosed-key certificate is issued by the Certificate Authority by adding information such as a term of validity to a disclosed key and a user ID supplied to the Certificate Authority as well as putting a digital signature of the Certificate Authority thereon. In accordance with the present embodiment, the license server 4 or the content server 3 issues a certificate and a secret key and, thus, also a disclosed key. Therefore, by presenting information such as a user ID and a password to the license server 4 to be cataloged therein, the user is capable of obtaining a disclosed-key certificate.

[0151] The disclosed-key certificate shown in Fig. 26 includes a message. The message includes a version number of the certificate, a serial number issued for the user of the certificate by the license server 4, an algorithm and parameters, which are used for a digital signature, the name of the Certificate Authority, the term of validity of the certificate, an ID assigned to the user of the certificate and a disclosed key of the certificate user. In this case, the Certificate Authority is the license server 4. The ID assigned to the user is a node ID or a leaf ID. A digital signature created by the license server 4 serving as the Certificate Authority is added to the message. The digital signature is data created by using a secret key of the license server 4 on the basis of a hash value generated by application of a hash function to the message.

[0152] In the case of the typical key organization shown in Fig. 12, for example, the node ID or the leaf ID is '0000' for device 0, '0001' for device 1 or '1111' for device 15. On the basis of such an ID, it is thus possible to determine which position in the tree structure, that is, which leaf or which node of the tree structure, a device (entity) identified by the ID is located at.

[0153] By distributing a license required in using a content separately from the content in this way, the content can be distributed with a higher degree of freedom. A content obtained by adopting an arbitrary method or obtained through an arbitrary route can thus be handled unitarily.

[0154] In addition, by constructing a file format as shown in Fig. 25, it is needless to say that the copyright of the content with such a format can be managed when the content is distributed through the Internet or even when the content is presented to SDMI (Secure Digital Music Initiative) equipment.

[0155] Furthermore, even if a content is presented by recording the content in a recording medium or presented through the Internet 2 as shown in Fig. 27, for example, by carrying out the same processing, it is possible to check out a content for typically a predetermined PD (Portable Device) used as SDMI (Secure Digital Music Initiative) equipment.

[0156] By referring to a flowchart shown in Fig. 28, the following description explains processing to check out a content for a client such a PD other than the client 1.

[0157] As shown in the figure, the flowchart begins with a step S191 at which the CPU 21 forms a judgment as to whether or not a digital signature has been put on the content. If the outcome of the judgment indicates that a digital signature has been put on the content, the flow of the processing goes on to a step S192 at which the CPU 21 extracts a disclosed-key certificate and carries out processing to authenticate the certificate by using a disclosed key of the license server 4 serving as the Certificate Authority. To put it in detail, the client 1 acquires a disclosed key for a secret key of the license server 4 from the license server 4, and uses the disclosed key to decrypt the digital signature put on the disclosed-key certificate. As described earlier by referring to Fig. 26, the digital signature is generated on the basis of the secret key of the license server 4 serving as the Certificate Authority, and can be decrypted by using the disclosed key of the license server 4. Furthermore, the CPU 21 applies a hash function to the whole message of the certificate to generate a hash value. Then, the CPU 21 compares the generated hash value with a hash value obtained as a result of decryption of the digital signature. If the generated hash value matches the hash value obtained as a result of decryption of the digital signature, the message is determined to be not a false message. If the generated hash value does not match the hash value obtained as a result of decryption of the digital signature, on the other hand, this certificate is determined to be a false one.

[0158] Thus, at the next step S193, the CPU 21 forms a judgment as to whether or not the certificate has been falsified. If the outcome of the judgment indicates that the certificate has not been falsified, the flow of the processing goes on to a step S194 at which the CPU 21 carries out processing to authenticate the certificate by using an EKB. The certificate is authenticated by examining whether or not the EKB can be traced on the basis of a leaf ID included in the certificate. For more information on the leaf ID, refer to Fig. 26. The authentication is explained by referring to Figs. 29 and 30 as follows.

[0159] As shown in Fig. 29, assume that a device owning a leaf key K1001 is a revoked device. In this case, an EKB having a tag and data (an encrypted key) like one shown in Fig. 30 are distributed to devices each corresponding to a leaf. In order to revoke a device '1001' shown in Fig. 29, this EKB is an EKB for updating keys KR, K1, K10 and K100.

[0160] All leaves other than a leaf corresponding to the revoked device '1001' are capable of acquiring the updated root key K(t)R. That is to say, since any of those leaves on a hierarchical layer below the node key K0 has the unupdated node key K0 inside the device, the leaf is capable of obtaining the updated root key K(t)R by decrypting the encrypted key Enc(K0, K(t)R) by using the key K0.

[0161] In addition, a leaf on a hierarchical layer under the node key K11 is capable of obtaining the updated node key K(t)1 by decryption of Enc(K11, K(t)1) by using the unupdated node key K11. Furthermore, the updated root key K(t)R can be obtained by decryption of Enc(K(t)1, K(t)R) by using the updated noide key K(t)1. By the same token, a leaf on a hierarchical layer under the node key K101 is also capable of obtaining the updated root key K(t)R.

[0162] In addition, a device '1000' owning an unrevoked leaf key K1000 is capable of obtaining a node key K(t)100 by decryption of Enc(K1000, K(t)100) by using its own leaf key K1000. The device then uses the node key K(t)100 to decrypt node keys on upper hierarchical layers sequentially, one key after another, to eventually obtain the updated root key K(t)R.

[0163] On the other hand, since a revoked device '1001' is not capable of obtaining an updated node key K(t)100 on an upper hierarchical layer one level above its own leaf by carrying out the EKB processing, the device is therefore incapable of obtaining the updated root key K(t)R eventually.

[0164] An authorized device, that is, the client 1, which was not revoked, receives an EKB with tags and data shown in Fig. 30 from the license server 4 and stores them therein.

[0165] Thus, each client is capable of carrying out an EKB tracing process by using the tags. The EKB tracing process is a process to form a judgment as to whether or not the key distribution tree can be traced from the root key at the top.

[0166] Assume that a leaf ID of '1001' assigned to a leaf '1001' shown in Fig. 29 is grasped as 4 bits, namely, '1', '0', '0' and '1'. The EKB tracing process is carried out to form a judgment as to whether the tree can be traced by examining bits starting with the most significant bit down through least significant bits sequentially, one bit after another. To be more specific, a '1' bit indicates that the tracing should go the right side while a '0' bit indicates that the tracing should go the left side.

[0167] Since the most significant bit of the ID '1001' is '1', the tracing goes on from the root key KR shown in Fig. 29 to the right side. The first tag of the EKB, that is, the tag having a number of 0, is 0 : {0, 0}, which indicates that data exists at both the branches. In this case, since the tracing is capable of going on to the right side, it is possible to reach the node key K1.

[0168] Next, the tracing goes on to a node on a hierarchical layer below the node key K1. Since the second bit of the ID '1001' is '0', the tracing goes on to the left side. The tag with a number of 1 indicates whether or not data exists on a hierarchical layer below a node key K0 on the left side. A tag indicating whether or not data exists on a hierarchical layer below the node key K1 is a tag with a number of 2. As shown in Fig. 30, the tag with a number of 2 is 2 : {0, 0}, which indicates that data exists at both the branches. Thus, the tracing goes on to the left side and is capable of reaching a node key K10.

[0169] Furthermore, since the third bit of the ID '1001' is 0, the tracing goes on to the left side. At that time, a tag indicating whether or not data exists on a hierarchical layer below the node key K10 is a tag with a number of 3. The tag with a number of 3 is 3 : {0, 0}, which indicates that data exists at both the branches. Thus, the tracing goes on to the left side and is capable of reaching a node key K100.

[0170] Furthermore, since the least significant bit of the ID '1001' is 1, the tracing goes on to the right side. A tag with a number of 4 corresponds to the node key K11. A tag indicating whether or not data exists on a hierarchical layer below the node key K100 is a tag with a number of 5. The tag with a number of 5 is 5 : {0, 1}, which indicates that no data exists on the right side. As a result, since the node '1001' can not be reached, the device with the ID of '1001' is determined to be a device incapable of acquiring the updated root key by using the EKB, or a revoked device.

[0171] On the other hand, for example, a device ID having a leaf key K1000 is '1000'. Thus, when the EKB tracing process based on tags in the EKB is carried out as described above, it is possible to reach the node '1000'. As a result, the device with the ID of '1000' is determined to be an authorized device.

[0172] Refer back to Fig. 28. At the next step S195, the CPU 21 forms a judgment as to whether or not the certificate has been revoked on the basis of a result of the authentication processing carried out at the step S194. If the certificate has not been revoked, the flow of the processing goes on to a step S196 at which processing is carried out to authenticate the digital signature by using a disclosed key included in the certificate.

[0173] That is to say, as shown in Fig. 26, the certificate includes a disclosed key of the certificate user (or the content author). The disclosed key is used for authenticating a digital signature Sig(Header) shown in Fig. 25. To put it in detail, the disclosed key is used for decrypting the digital signature Sig(Header) in order to produce a hash value. This hash value is compared with a hash value obtained by application of a hash function to a header shown in Fig. 25. If both the hash values

match each other, the header is confirmed as a header, which has not been falsified. Otherwise, the header is determined to have been falsified.

[0174] Then, at the next step S197, the CPU 21 forms a judgment as to whether or not the header has been falsified. If the header has not been falsified, the flow of the processing goes on to a step S198 at which the watermark is authenticated. Subsequently, at the next step S199, the CPU 21 forms a judgment as to whether or not a result of the authentication of the watermark indicates that a check-out is possible. If a check-out is possible, the flow of the processing goes on to a step S200 at which the CPU 21 carries out the check out. To put it concretely, the CPU 21 transfers a content to the client 1 serving as a check-out destination to be copied thereby.

[0175] If the outcome of the judgment formed at the step S191 indicates that the digital signature does not exist, the outcome of the judgment formed at the step S193 indicates that the certificate has been falsified, the outcome of the judgment formed at the step S195 indicates that the certificate cannot be authenticated by using the EKB, the outcome of the judgment formed at the step S197 indicates that a result of the authentication of the digital signature indicates that a header has been falsified or the outcome of the judgment formed at the step S199 indicates that the watermark includes a description inhibiting the check-out, on the other hand, the flow of the processing goes on to a step S201 at which an error-handing process is carried out. That is to say, in this case, the check-out is prohibited.

[0176] As described above, a certificate and a secret key are distributed from the license server 4 to the user. By adding a digital signature at creation of a content, the genuineness of the author of the content can be assured. As a result, illegal distribution of the content can be avoided.

[0177] In addition, by detecting a watermark at the creation of a content and adding the watermark to the digital signature, falsification of the watermark can be avoided. Thus, the genuineness of the content can be assured.

[0178] As a result, once created, the genuineness of the original content can be assured without regard to what format the content is distributed in.

[0179] In addition, a content does not have usage conditions. Instead, usage conditions are added to a license for the content. Thus, by changing usage conditions included in the license, conditions for using the content are also modified as well.

[0180] Next, a method of using a mark is explained. In accordance with the present embodiment, usage conditions are added to not a content but a license for the content as described above. However, usage circumstances may vary from content to content. In order to solve this problem, a mark is added to a content in accordance with the present embodiment as shown in Fig. 25.

[0181] Since a license is associated with a plurality of contents, it is difficult to describe usage circumstances of each content in only usage conditions included in the license. In order to solve this problem, by adding usage circumstances to the content, it is possible to manage the individual contents while managing the license.

[0182] As shown in Fig. 31, the mark typically includes an ID (leaf ID) assigned to the user, an ownership flag, a usage start time and a copy count.

[0183] In addition, the mark also includes an additional digital signature created on the basis of a message such as the leaf ID, the ownership flag, the usage start time and the copy count.

[0184] The ownership flag is added, for example, when the user buys a license, which allows a content to be used only for a predetermined period of time, as it is, or when the usage period is changed to a permanent usage period. The usage start time is described when the use of the content is started within a predetermined period of time. Assume that the period to download the content is limited. In this case, if the content is downloaded within the limited period of time, the date and time at which the content is actually downloaded are recorded as the usage start time. In this way, legal use of the content within a period of time is proven.

[0185] Recorded as a log, the copy count is the number of operations carried out so far to copy the contents.

[0186] By referring to a flowchart shown in Fig. 32, the following description explains processing carried out to add a mark to a content when the user purchases a license.

[0187] As shown in the figure, the flowchart begins with a step S221 at which the CPU 21 makes an access to the license server 4 through the Internet 2 in accordance with a command entered by the user via the input unit 26.

[0188] Then, at the next step S222, the CPU 21 fetches an input entered by the user through the input unit 26 and transmits a request to purchase a license according to the input to the license server 4.

[0189] In response to this request, the license server 4 presents a price to purchase the license at a step S242 of a flowchart shown in Fig. 33 as will be described later by referring to the flowchart shown in Fig. 33. Subsequently, at the next step S223, the CPU 21 employed in the client 1 receives the price transmitted by the license server 4, and displays the price on the output unit 27.

[0190] On the basis of the displayed price, the user forms a judgment as to whether to agree or disagree on the price. The user enters the outcome of the judgment to the input unit 26.

[0191] Then, at the next step S224, on the basis of the judgment outcome entered to the input unit 26, the CPU 21 forms a judgment as to whether the price has been agreed or disagreed on. If the price has been agreed on, the flow of the processing goes on to a step S225 at which the CPU 21 carries out processing to no-

tify the license server 4 that the price has been agreed on.

[0192] Receiving this notification, the license server 4 transmits information representing a license purchase at the price, that is, a mark including a described ownership flag at a step S244 of the flowchart shown in Fig. 33. Subsequently, at the next step S226, the CPU 21 employed in the client 1 receives the mark transmitted by the license server 4. Then, at the next step S227, the CPU 21 carries out processing to embed the mark in the content. Thus, the mark including a described ownership flag as shown in Fig. 31 is recorded in the content associated with the purchased license as a mark for the content. In addition, since the message is updated at that time, the CPU 21 also updates the digital signature shown in Fig. 25 and stores the updated digital signature in the recording medium.

[0193] If the outcome of the judgment formed at the step S224 indicates that the price presented by the license server 4 has been disagreed on, on the other hand, the flow of the processing goes on to a step S228 at which the CPU 21 notifies the license server that the price has been disagreed on.

[0194] For the processing carried out by the client 1 as described above, the license server 4 performs processing represented by the flowchart shown in Fig. 33.

[0195] As shown in the figure, the flowchart begins with a step S241 at which the CPU 21 employed in the license server 4 receives a request for a purchase of a license from the client 1. As described above, such a request is transmitted by the client 1 at the step S222 of the flowchart shown in Fig. 32. Then, at the next step S242, the CPU 21 reads out the price of the license to be purchased by the user from the storage unit 28, and transmits the price to the client 1.

[0196] As described above, in response to the disclosed price, the client 1 transmits the outcome of the judgment as to whether the price is agreed or disagreed on.

[0197] Subsequently, at the next step S243, the CPU 21 employed in the license server 4 determines whether the price is agreed or disagreed on by the client 1 on the basis of the judgment's outcome received from the client 1. If the price is agreed on, the flow of the processing goes on to a step S244 to generate a mark including a message representing a purchase of a license for the content, put a digital signature on the mark by using a secret key of its own and transmit the mark to the client 1. As described above, the mark transmitted in this way is recorded on the content in the storage unit 28 employed in the client 1 at the step S227 of Fig. 32.

[0198] If the CPU 21 employed in the license server 4 determines that the price is disagreed on by the client 1 at this step S243, on the other hand, the processing of the step S244 is skipped. That is to say, in this case, the processing to purchase a license is not carried out eventually. Thus, no mark is transmitted to the client 1.

[0199] Fig. 34 is a diagram showing a typical configuration of the mark transmitted from the license server 4 to the client 1 at the step S244. In this typical configuration, the mark comprises the leaf ID of the user, an ownership flag (Own) and a digital signature Sigs(Leaf ID, Own), which is generated from the leaf ID and the ownership flag on the basis of a secret key S of the license server 4.

[0200] It should be noted that the mark is valid only for a specific content issued to a specific user. Thus, if the specific content is copied, the mark in the copied content is invalid.

[0201] In this way, even if a content is separated from a license and usage conditions are associated with the license, it is possible to render services according to usage circumstances for individual contents.

[0202] Next, grouping is explained. A plurality of apparatuses and mediums are collected in a group, in which a content can be exchanged with a high degree of freedom. The formation of such a group is referred to as grouping. Normally, the grouping forms a group comprising apparatuses and mediums, which are owned by an individual. Conventionally, the grouping also includes an operation to set a group key for each group. By associating a plurality of apparatuses and mediums collected in a group with a common license, however, the grouping can be done with ease.

[0203] In addition, the grouping can be carried out by cataloging the apparatuses in advance. This kind of grouping is explained as follows.

[0204] In this case, the user needs to catalog certificates of apparatuses to be grouped in a server in advance. The processing to catalog such certificates is explained by referring to flowcharts shown in Figs. 35 and 36.

[0205] First of all, the processing to catalog the certificate of a client, that is, an apparatus to be grouped, is explained by referring to the flowchart shown in Fig. 35. As shown in the figure, the flowchart begins with a step S261 at which the CPU 21 employed in the client 1 creates its own certificate as a certificate of an apparatus to be grouped. This certificate includes its own disclosed key.

[0206] Then, at the next step S262, the CPU 21 makes an access to the content server 3 on the basis of an input entered by the user to the input unit 26. Subsequently, at the next step S263, the certificate created at the step S261 is transmitted to the content server 3.

[0207] It should be noted that a certificate received from the license server 4 can also be used as it is as the certificate described above.

[0208] The processing described above is carried out by all apparatuses to be grouped.

[0209] By referring to the flowchart shown in Fig. 36, the following description explains processing carried out by the content server 3 to catalog a certificate created by the processing carried out by the client 1 to catalog the certificate as represented by the flowchart shown in

Fig. 35.

[0210] As shown in the figure, the flowchart begins with a step S271 at which the CPU 21 employed in the content server 3 receives a certificate from the client 1. Then, at the next step S272, the certificate is cataloged in the storage unit 28.

[0211] The processing described above is carried out for each apparatus to be grouped. As a result, certificates of devices composing each group are cataloged in the storage unit 28 employed in the content server 3 as shown in Fig. 37.

[0212] In the example shown in Fig. 37, certificates C11 to C14 are cataloged as certificates of group 1. These certificates C11 to C14 include corresponding disclosed keys K_{P11} to K_{P14} respectively.

[0213] By the same token, certificates C21 to C23 are cataloged as certificates of group 2. These certificates C21 to C23 include corresponding disclosed keys K_{P21} to K_{P23} respectively.

[0214] With a certificate cataloged for each apparatus composing such a group, the content server 3 carries out processing represented by a flowchart shown in Fig. 38 when the user of an apparatus pertaining to a group makes a request for presentation of a content.

[0215] As shown in the figure, the flowchart begins with a step S281 at which the CPU 21 employed in the content server 3 carries out processing to authenticate the group's certificate selected among the ones cataloged in the storage unit 28.

[0216] As explained earlier by referring to Figs. 29 and 30, in this authentication processing, an EKB is traced by using tags on the basis of the apparatus' leaf ID included in the certificate. The EKB has been distributed by the license server 4 to the content server 3. The authentication processing eliminates a revoked certificate.

[0217] Then, at the next step S282, the CPU 21 employed in the content server 3 selects a certificate determined to be valid as a result of the authentication processing carried out at the step S281. Subsequently, at the next step S283, the CPU 21 encrypts a content key by using a disclosed key of the apparatus' certificate selected in the processing carried out at the step S282. Then, at the next step S284, the CPU 21 transmits the content key encrypted in the processing carried out at the step S283 along with its content to the group's apparatus making the request for presentation of the content.

[0218] Assume that the certificate C14 of one of the groups shown in Fig. 37 has been revoked. In this case, in the processing carried out at the step S283, encrypted data shown in Fig. 39 is typically generated.

[0219] To put in detail, in the encrypted data shown in Fig. 39, a content key Kc has been encrypted by using a disclosed key K_{P11} of the certificate C11, a disclosed key K_{P12} of the certificate C12 or a disclosed key K_{P13} of the certificate C13.

[0220] When receiving the content from the license server 3 as a result of the processing represented by the flowchart shown in Fig. 38, the apparatus or the client pertaining to the group carries out processing represented by a flowchart shown in Fig. 40.

[0221] As shown in Fig. 40, the flowchart begins with a step S291 at which the CPU 21 employed in the client 1 receives the content key Kc and the content, which are transmitted by the content server 3 in the processing carried out at the step S284 of the flowchart shown in Fig. 38. The content has been encrypted by using the content key Kc, which has been encrypted by using a disclosed key held by the apparatus as described above. (Refer to Fig. 39).

[0222] Then, at the next step S292, the CPU 21 decrypts the content key Kc, which has been received in the processing carried out at the step S291 and is destined for the client 1, by using a secret key owned by the client 1. The CPU 21 then uses the decrypted content key to decrypt the content.

[0223] For instance, take the apparatus corresponding to the certificate C11 shown in Fig. 39 as an example. The apparatus decrypts the content key Kc by using its own secret key corresponding to the disclosed key K_{P11}. The apparatus then uses the decrypted content key Kc to decrypt the content.

[0224] The same processing is carried out for apparatuses associated with the certificates C12 and C13. An apparatus associated with the revoked certificate C14 does not receive the content key Kc encrypted by its disclosed key attached to the content. Thus, the apparatus is not capable of decrypting the content key Kc by using its own secret key and thus incapable of decrypting the content by using the decrypted content key Kc.

[0225] As described above, apparatuses are grouped with respect to content keys, that is, contents. However, apparatuses are grouped with respect to license keys, that is, licenses.

[0226] As described above, apparatuses can be grouped without using special group keys or ICVs (Integrity Check Values) to be described later. This kind of grouping is suitable for a group with a small scale.

[0227] In accordance with the present embodiment, a license can be checked out, checked in, moved and copied. However, these operations must be based on rules determined by the SDMI.

[0228] By referring to flowcharts shown in Figs. 41 and 42, the following description explains processing to check out a license by using such a client.

[0229] The description begins with an explanation of processing carried out by a client to check out a license to another client with reference to the flowchart shown in Fig. 41. As shown in the figure, the flowchart begins with a step S301 at which the CPU 21 employed in the client 1 reads out the number of check-out operations (N1) for a license to be checked out. The number of check-out operations (N1) is included in usage conditions shown in Fig. 8. Thus, the number of check-out operations (N1) is read out from the usage conditions.

[0230] Then, at the next step S302, the CPU 21 employed in the client 1 reads out the maximum number of check-out operations (N2) for a license to be checked out. Also in this case, the maximum number of check-out operations (N2) is read out from the usage conditions.

[0231] Subsequently, at the next step S303, the CPU 21 compares the number of check-out operations (N1) read out at the step S301 with the maximum number of check-out operations (N2) read out at the step S302 to form a judgment as to whether the number of check-out operations (N1) is greater or smaller than the maximum number of check-out operations (N2).

[0232] If the number of check-out operations (N1) is found smaller than the maximum number of check-out operations (N2), the flow of the processing goes on to a step S304 at which the CPU 21 acquires the leaf key of a partner apparatus from the partner apparatus, which is a client serving as a check-out destination. The acquired leaf key is cataloged on a check-out list stored in the storage unit 28, being associated with a license ID serving as a check-out object.

[0233] Then, at the next step S305, the CPU 21 increments the number of check-out operations (N1) read out at the step S301 by 1. Subsequently, at the next step S306, the CPU 21 finds an ICV based on the message of the license. The ICV will be described later by referring to Figs. 46 to 50. By using the ICV, it is possible to prevent the license from being falsified.

[0234] Then, at the next step S307, the CPU 21 encrypts the license serving as the check-out object and the ICV found at the step S306 by using the disclosed key owned by the client 1 itself. The encrypted license and the encrypted ICV are transmitted to the partner apparatus to be copied thereby along with an EKB and a certificate. Subsequently, at the next step S308, the CPU 21 catalogs the ICV found at the step S306 on the check-out list stored in the storage unit 28 by associating the ICV with the license ID and the leaf key of the partner apparatus.

[0235] If the outcome of the judgment formed at the step S303 indicates that the number of check-out operations (N1) is not smaller than (for example, equal to) the maximum number of check-out operations (N2), on the other hand, the flow of the processing goes on to a step S309 at which the CPU 21 carries out an error-handling process. This is because, since the number of check-out operations (N1) is not smaller than the maximum number of check-out operations (N2), indicating that the license has been checked out as many times as the number of allowable check-out operations (N2), the license can no longer be checked out. Thus, in this case, the license is not checked out.

[0236] By referring to the flowchart shown in Fig. 42, the following description explains processing carried out by a client receiving a license checked out in the check-out processing represented by the flowchart shown in Fig. 41.

[0237] The flowchart shown in Fig. 42 begins with a step S321 at which the CPU 21 employed in the client transmits the leaf key owned by the client itself to the partner apparatus, that is, the client 1 checking out a license. The leaf key is stored in the partner apparatus at the step S304, being associated with a license ID.

[0238] Then, at the next step S322, the CPU 21 receives an encrypted license and an encrypted ICV along with an EKB and a certificate from the partner client 1. As described earlier, the partner client 1 transmits the encrypted license and the encrypted ICV along with the EKB and the certificate at the step S307 of the flowchart shown in Fig. 41.

[0239] Subsequently, at the next step S323, the CPU 21 stores the encrypted license, the encrypted ICV, the EKB and the certificate, which were received at the step S322, in the storage unit 28.

[0240] The client 1 receiving a checked-out license as described above uses the checked-out license to play back a content in accordance with processing represented by a flowchart shown in Fig. 43.

[0241] As shown in the figure, the flowchart begins with a step S341 at which the CPU 21 employed in the client 1 finds an ICV of a content specified by a command entered by the user to the input unit 26 as a content to be played back. Subsequently, at the next step S342, the CPU 21 decrypts an encrypted ICV stored in the storage unit 28 by using a disclosed key included in the certificate.

[0242] Then, at the next step S343, the CPU 21 forms a judgment as to whether or not the ICV found at the step S341 matches the ICV read out and decrypted in the processing carried out at the step S342. The former matching the latter indicates that the license has not been falsified. In this case, the flow of the processing goes on to a step S344 at which the CPU 21 carries out processing to play back the content.

[0243] If the outcome of the judgment formed at the step S343 indicates that the two ICVs do not match each other, it is feared that the license has been falsified, on the other hand. In this case the flow of the processing goes on to a step S345 at which the CPU 21 carries out an error-handling process. That is to say, the content cannot be played back by using this license.

[0244] By referring to a flowchart shown in Fig. 44, the following description explains processing carried out by a client to check in a license, which was once checked out to another client 1 as described above.

[0245] As shown in the figure, the flowchart begins with a step S361 at which the CPU 21 employed in the client receives the leaf key of a partner apparatus and the ID of a license to be checked in. The partner apparatus is a client 1, which returns or checks in a license. Then, at the next step S362, the CPU 21 forms a judgment as to whether or not the license to be checked in, which is obtained at the step S361, is a license checked out by the client itself to the partner apparatus. This judgment is based on the ICV, the leaf key and the license

ID, which were stored in the storage unit 28 in the processing carried out at the step S308 of the flowchart shown in Fig. 41. That is to say, the CPU 21 determines whether the ICV, the leaf key and the license ID, which were received at the step S361, have been cataloged on the check-out list stored in the storage unit 28. If they have been cataloged on the check-out list, the CPU 21 determines that the license to be checked in is a license checked out by the client itself to the partner apparatus.

[0246] If the license to be checked in is a license checked out by the client itself to the partner apparatus, the flow of the processing goes on to a step S363 at which the CPU 21 makes a request for deletion of the EKB, the certificate and the license of the partner apparatus. As will be described later, the partner apparatus deletes the license, the EKB and the certificate at a step S383 of a flowchart shown in Fig. 45 in accordance with the request.

[0247] Then, at the next step S364, since a check-out license is checked in, the CPU 21 decrements the number of check-out operations (N1) by 1.

[0248] Subsequently, at the next step S365, the CPU 21 forms a judgment as to whether or not another license has been checked out to the partner apparatus. If another license checked out to the partner apparatus does not exist, the flow of the processing goes on to a step S366 at which the CPU 21 deletes the partner apparatus from the check-out list for cataloging the partner apparatus as a check-in-partner apparatus. If the outcome of the judgment formed at the step S365 indicates that another license checked out to the partner apparatus exists, on the other hand, the processing of the step S366 is skipped. This is because it is quite within the bound of possibility that the other license is checked in by the partner apparatus.

[0249] If the outcome of the judgment formed at the step S362 indicates that the license to be checked in is not a license checked out by the client itself to the partner apparatus, on the other hand, the flow of the processing goes on to a step S367 at which the CPU 21 carries out an error-handling process. That is to say, in this case, the check-in processing is not carried out since the license is not a license managed by the client itself.

[0250] In an attempt made by the user to illegally copy the license, the check-in processing cannot be carried out since the stored ICV is not equal to the ICV found on the basis of the license acquired in the processing carried out at the step S361.

[0251] Fig. 45 is a flowchart representing processing carried out by a client 1 issuing a request for the processing to check in a license to another client carrying out the license-check-in processing represented by the flowchart shown in Fig. 44.

[0252] The flowchart shown in Fig. 45 begins with a step S381 at which the CPU 21 employed in the client 1 transmits a leaf key and the ID of a license to be checked in to a partner apparatus, which is the client 1 carrying out the license-check-in processing represented by the flowchart shown in Fig. 44. As described above, the partner apparatus receives the leaf key and the license ID at the step S361 and carries out processing to authenticate the license to be checked in on the basis of the leaf key and the license ID at the step S362.

[0253] Then, at the next step S382, the CPU 21 employed in the client 1 forms a judgment as to whether or not a request for deletion of the license has been received from the partner apparatus. As described earlier, if the license is a proper license to be checked in, the partner apparatus makes a request for deletion of the license, the EKB and the certificate in the processing carried out at the step S363. If the outcome of the judgment formed at the step S382 indicates that such a request has been received, the flow of the processing goes on to a step S383 at which the CPU 21 deletes the license, the EKB and the certificate. That is to say, the client 1 thus becomes no longer capable of using the license. Since the number of check-out operations (N1) is decremented by 1 by the partner apparatus in the processing carried out at the step S364 of the flowchart shown in Fig. 44, the check-in processing is ended.

[0254] If the outcome of the judgment formed at the step S382 indicates that such a request was not received, on the other hand, the flow of the processing goes on to a step S384 at which the CPU 21 carries out an error-handling process. That is to say, in this case, the check-in processing cannot be carried out due to some reasons such as a discrepancy in ICV.

[0255] The check-out processing and the check-in processing have been explained so far. Processing to copy or move a license can also be carried out as well.

[0256] The following description explains processing to generate an ICV (Integrity Check Value) of a license, associate the ICV with the license and form a judgment as to whether or not the license has been falsified by computation of an ICV in order to prevent the license from being falsified. It should be noted that the same processing can be applied to a content.

[0257] An ICV (Integrity Check Value) of a license is computed by typically application of a hash function to the license as follows:

$$ICV = hash(Kicv, L1, L2, \cdots)$$

where notation Kicv denotes an ICV generation key whereas symbols L1 and L2 each denote information on the license. A MAC (Message Authentication Code) of important information of the license is used as the information represented by L1 and L2.

[0258] Fig. 46 is a diagram showing typical generation of a MAC value by using a DES encryption processing configuration. As is obvious from the configuration shown in Fig. 46, a processed message is divided into 8-byte units. In the following description, the divided message is referred to as M1, M2, ··· and MN. First of

all, an initial value IV and M1 are supplied to a processing unit 24-1A for carrying out exclusive logical sum processing to result in an exclusive logical sum I1. Then, the exclusive logical sum I1 is supplied to a DES encryption unit 24-1B for encrypting 11 by using a key K1 to produce an encryption result E1. Subsequently, E1 and M2 are supplied to a processing unit 24-2A for carrying out exclusive logical sum processing to result in an exclusive logical sum 12. Then, the exclusive logical sum 12 is supplied to a DES encryption unit 24-2B for encrypting 12 by using a key K1 to produce an encryption result E2. Thereafter, these operations are carried out repeatedly to encrypt all the messages. Eventually, a result EN generated by a DES encryption unit 24-NB is a MAC (Message Authentication Code).

[0259] A hash function is then applied to such a license MAC value and an ICV generation key to generate an ICV (Integrity Check Value). For example, an ICV computed at generation of a license is compared with an ICV newly calculated from a license. If the ICVs match each other, the license is assured not to have been falsified. If the ICVs do not match each other, on the other hand, the license is determined to have been falsified.

[0260] The following description explains a configuration to use an EKB (Enabling Key Block) for transmitting a key Kicv for generating an ICV (Integrity Check Value) of a license. In the configuration, message data encrypted by using the EKB is used as the key Kicv for generating an ICV (Integrity Check Value) of a license.

[0261] To put it concretely, Figs. 47 and 48 are each a diagram showing a typical configuration of using an EKB (Enabling Key Block) to distribute a key Kicv for generating a common license's ICV (Integrity Check Value) for forming a judgment as to whether or not the license has been falsified when transmitting the license to a plurality of devices. To be more specific, Fig. 47 is a diagram showing typical distribution of a decryptable key Kicv for generating a license's ICV (Integrity Check Value) to devices 0, 1, 2 and 3. On the other hand, Fig. 48 is a diagram showing typical distribution of a decryptable key Kicv for generating a license's ICV (Integrity Check Value) to devices 0, 1, and 2 only but not to device 3, which has been revoked.

[0262] In the typical distribution shown in Fig. 47, an encrypted EKB (Enabling Key Block), which can be decrypted, is generated. The EKB is used for transmitting data Enc(K(t)00, Kicv) and an updated node key K(t)00 to devices 0, 1, 2 and 3. The data Enc(K(t)00, Kicv) is a result of encryption of the check-value generation key Kicv by using the updated node key K(t)00. The node key K(t)00 has been updated by using a node key and a leaf key, which are owned by each of devices 0, 1, 2 and 3. As shown on the right side of Fig. 47, first of all, each of devices 0, 1, 2 and 3 decrypts the EKB to obtain the updated node key K(t)00. Then, the updated node key K(t)00 is used for decrypting the encrypted check-value generation key Enc(K(t)00, Kicv) to obtain the check-value generation key Kicv.

[0263] Other devices 4, 5, 6, 7 and so on are each incapable of obtaining the updated node key K(t)00 by processing an EKB (Enabling Key Block) and by using a node key and a leaf key, which are owned by each of devices, even if the EKB is received by the devices. Thus, the check-value generation key can be transmitted to only authorized devices with a high degree of safety.

[0264] Fig. 48 is a diagram showing a case in which device 3 pertaining to a group enclosed by a dotted line in Fig. 12 has been revoked because, for example, a key has leaked out, so that an EKB (Enabling Key Block) is generated and distributed to only other members of the group, namely, devices 0, 1 and 2. The EKB (Enabling Key Block) can be decrypted only by the device 0, 1 and 2. The EKB (Enabling Key Block) shown in Fig. 48 and data Enc(K(t)00, Kicv) are distributed. As described earlier, the data Enc(K(t)00, Kicv) is a result of encryption of a check-value generation key Kicv by using a node key K(t)00.

[0265] On the right side of Fig. 48, a decryption procedure is shown. As shown in the figure, first of all, devices 0, 1 and 2 each acquire the updated node key K(t)00 by carrying out processing to decrypt the received EKB (Enabling Key Block) by using a leaf key or a node key owned by itself. Then, the check-value generation key Kicv is obtained by decryption based on the updated node key K(t)00.

[0266] Other devices 4, 5, 6 and so on of the group shown in Fig. 12 are each incapable of acquiring the updated node key K(t)00 by using its own leaf key and node key even if the same EKB (Enabling Key Block) is distributed to those other devices. By the same token, revoked device 3 is also incapable of acquiring the updated node key K(t)00 by using its own leaf key and node key even if the same EKB (Enabling Key Block) is distributed to this device. Thus, only an authorized device is capable of decrypting and using the check-value generation key Kicv.

[0267] In this way, by utilizing distribution of the check-value generation key Kicv through the use of an EKB, the amount of distributed data can be reduced and it is possible to safely distribute the check-value generation key Kicv to only authorized parties capable of decrypting the check-value generation key Kicv.

[0268] By using such an ICV (Integrity Check Value) of a license, it is possible to avoid illegal copies of the EKB and the encrypted license. Assume that media 1 is used for storing licenses L1 and L2 along with an EKB (Enabling Key Block) that can be used for acquiring their license keys as shown in Fig. 49A. Let what is stored in media 1 be copied to media 2. In this case, the EKB and the licenses can be copied. A device capable of decrypting the EKB will also be capable of using the licenses.

[0269] In a configuration shown in Fig. 49B, an integrity check value ICV(L1, L2) is stored in each media, being associated with licenses also stored therein. It

should be noted that ICV(L1, L2) is an integrity check value of licenses L1 and L2 and is computed by applying a hash function to licenses L1 and L2 as follows:

$$ICV = hash(Kicv, L1, L2)$$

[0270] In the configuration shown in Fig. 49B, information stored in media 1 include licenses 1 and 2 as well as the integrity check value ICV(L1, L2), which is computed by applying a hash function to licenses L1 and L2. On the other hand, information stored in media 2 includes license 1 and an integrity check value ICV(L1), which is computed by applying a hash function to license L1.

[0271] In this configuration, assume that {EKB, license 2} is copied from media 1 to media 2. In this case, a new license check value ICV(L1, L2) can be generated in media 2. The new license check value ICV(L1, L2) is different from Kicv(L1) stored in media 2. It is thus obvious that the new license check value ICV(L1, L2) can be used to store a new license in media 2 by falsification or an illegal copy operation. In a device for playing back information stored in media 2, however, generated and stored ICVs can be checked at a step prior to a playback step to form a judgment as to whether the ICVs match each other. If the generated ICV is determined to be an ICV not matching the stored ICV, no playback operation is carried out. In this way, in this configuration, it is possible to prevent the license obtained by falsification or by carrying out an illegal copy operation from being played back.

[0272] In addition, in order to further enhance the degree of safety, it is possible to devise a configuration in which the ICV (Integrity Check Value) of a license is generated on the basis of data including the value of a writable counter. To put it concretely, in the configuration, the ICV (Integrity Check Value) of a license is computed as follows:

$$ICV = hash(Kicv, counter + 1, L1, L2, \cdots)$$

where notation (counter + 1) indicates that the value of the counter is incremented by 1 each time the ICV is updated. It should be noted that the value of the counter needs to be stored in a secure memory in this configuration.

[0273] Moreover, in a configuration wherein the ICV (Integrity Check Value) of a license cannot be stored in the same media as the license, the ICV (Integrity Check Value) of the license may be stored in a media different from the media for storing the license.

[0274] Assume that a license is stored in a media with no protection against an illegal copy operation. Examples of such a media are a read-only memory and an ordinary M0 disk In this case, if an ICV (Integrity Check Value) is also stored in the same media, it is quite within the bounds of possibility that an unauthorized user is capable of updating the ICV. It is thus feared that the safety of the ICV is not assured. In order to solve this problem, the ICV is stored in a safe media of the host machine and used for controlling operations to copy the license. Examples of the copy operation are operations to check in, check out and move the license. In such a configuration, it is thus possible to execute safety management of the ICV and check falsification of the license.

[0275] Fig. 50 is a diagram showing a typical configuration implementing the scheme described above. In the typical configuration shown in Fig. 50, a media 2201 with no protection against an illegal copy operation is used for storing licenses 1 to 3. Examples of the media 2201 are a read-only memory and an ordinary M0 disk On the other hand, an ICV (Integrity Check Value) for these licenses is stored in a safe media 2202 employed in a host machine to which the user is not allowed to make an access with a high degree of freedom. Thus, in this typical configuration, the user is prevented from illegally updating the ICV (Integrity Check Value). When a device on which the media 2201 is mounted plays back information from the media 2201, for example, a PC serving as the host machine of the device or a server may be configured to check ICVs for forming a judgment as to whether or not the media is allowed to play back. In such a configuration, it is thus possible to prevent an operation to play back an illegally copied or falsified license.

[0276] In addition, a client provided by the present invention can also be implemented by an apparatus other than the so-called personal computer. Examples of an apparatus other than the so-called personal computer are a PDA (Personal Digital Assistant), a cellular phone and a game terminal.

[0277] If the series of pieces of processing is implemented by software, a program composing the software can be installed from a network or a recording medium into a computer including embedded special hardware or into a computer of another type such as a general-purpose personal computer capable of carrying out a variety of functions by execution of various programs installed in the personal computer.

[0278] A recording medium provided separately from the main unit of the apparatus serving as a client or a server is distributed to users for presenting a program recorded in the medium to users. The recording medium can be a package medium. As shown in Fig. 2, examples of the package medium are the magnetic disk 41 including a floppy disk, the optical disk 42 including a CD-ROM (Compact-Disk Read-Only Memory) and a DVD (Digital Versatile/Video Disk), the magneto-optical disk 43 including an MD (Mini Disk) and the semiconductor memory 44. Instead of installing a program from a network or a recording medium, the program can be presented to a user by storing in advance the program in a recording medium embedded in the main unit of the apparatus. As shown in Fig. 2, examples of the embed-

ded recording medium are the ROM 22 and a hard disk included in the storage unit 28.

[0279] In this specification, steps describing a program stored in the recording medium can of course be executed sequentially one step after another in accordance with a written procedure. It should be noted, however, that the steps do not have to be executed sequentially but, instead, the steps may also include pieces of processing to be carried out in parallel or individually.

[0280] In addition, it is desirable to also encrypt a program executed to implement processing related to security in order to prevent the processing of the program itself from being analyzed. For example, a program of processing carried out to execute an encryption process can be designed as a tamper resistant module.

[0281] Furthermore, the information included in the header of a content to specify a license for allowing the use of the content does not have to be a license ID for uniquely identifying the license. In the embodiment described above, a license ID is information for specifying a license required in utilization of a content, information for specifying a content the use of which is allowed by a certain license and information for identifying a license requested by a license request from the client 1. Instead, a list of various kinds of attribute information related to a content may also be included in the content, and conditions of attribute of contents may also be included in a license for specifying contents allowed to be used. In this case, attribute information included in a content is information for specifying a license for allowing utilization of the content and information for specifying a content the use of which is allowed by a license in accordance with a condition equation included in the license. A license ID is information for uniquely identifying a license. In this way, a content can be associated with a plurality of licenses so that the content can be issued in a more flexible manner.

[0282] In addition, the technical term 'content-exchanging system' used in this specification means the entire system comprising a plurality of apparatus.

[0283] As described above, in accordance with the information-processing apparatus and method provided by the above described embodiments of the invention and the program for implementing the information-processing method, encrypted data can be distributed with a high degree of freedom and, by acquiring a license provided separately from a content, the user is capable of utilizing the content. As a result, a copyright can be protected and a proper usage fee can be collected without a hindrance to distribution of a content.

Claims

1. An information-processing apparatus for allowing usage of a content by requiring a license for using said content, said information-processing apparatus comprising:

content storage means for storing a license-specifying information for specifying said license required by said user in using said content, encrypted data of said content and key information required for decrypting said encrypted data of said content;

license storage means for storing said license including a content-specifying information for specifying said content, the use of which is allowed;

judgment means for forming a judgment as to whether or not said license required by said user in using said content has been stored in said license storage means; and

decryption means, which is used for decrypting said encrypted data of said content on condition that an outcome of said judgment formed by said judgment means indicates that said license required by said user in using said content has been stored in said license storage means.

2. An information-processing apparatus according to claim 1, further comprising:

transmission means for transmitting a license request including said license-identification information for identifying said license required by said user in using said content to a license server; and

receiving means for receiving said license transmitted by the license server,

wherein said license received by said receiving means is stored in said license storage means.

3. An information-processing apparatus according to claim 1, further comprising reproducing means for reproducing said content's data decrypted by said decryption means, wherein said data of said content is text data, image data, audio data, moving-picture data or a combination of them.

4. An information-processing apparatus according to claim 1, further comprising device-node-key storage means for storing a device node key, wherein:

said key information includes an EKB (Enabling Key Block); and

said decryption means decrypts said EKB (Enabling Key Block) by using said device node key stored in said device-node-key storage means, and decrypts data of said content by using a root key obtained as a result of decryption of said EKB.

5. An information-processing apparatus according to claim 4 wherein:

said key information further includes a content key encrypted by using said root key of said EKB (Enabling Key Block);
said data of said content is encrypted by using said content key; and
said decryption means decrypts said data of said content encrypted by using said content key by using said root key obtained as a result of decryption of said EKB (Enabling Key Block) by using said device node key stored in said device-node-key storage means.

6. An information-processing apparatus according to claim 1 wherein said license further includes usage-condition information showing a condition for using said content, use of which is allowed by said license.

7. An information-processing apparatus according to claim 1 wherein said license further includes an electronic signature signed by using a secret key of a license server.

8. An information-processing apparatus according to claim 2, further comprising a terminal-ID storage means for storing a terminal specifying information identifying said information-processing apparatus, wherein:

said license request transmitted by said transmission means further includes said terminal-identification information stored in said terminal-ID storage means;
said license received by said receiving means includes a terminal ID; and
said judgment means compares said terminal-identification information included in said license with said terminal-identification information stored in said terminal-ID storage means, and determines that said license received by said receiving means is a license allowing use of said content only if said terminal-identification information included in said license matches said terminal-identification information stored in said terminal-ID storage means.

9. An information-processing method for allowing a user to use a content by requiring said user to have a license for using said content, said information-processing method comprising:

a content storage step of storing a license-specifying information for specifying said license required by said user in using said content, encrypted data of said content and key information required for decrypting said encrypted data of said content;
a license storage step of storing said license including a content-specifying information for specifying said content, the use of which is allowed by said license;
a judgment step of forming a judgment as to whether or not said license required by said user in using said content has been stored in said license storage means; and
a decryption step, at which said encrypted data of said content is decrypted on condition that an outcome of said judgment formed at said judgment means indicates that said license required by said user in using said content has been stored in said license storage means.

10. A program to be executed by a computer for carrying out processing of allowing a user to use a content by requiring said user to have a license for using said content, said program comprising:

a content storage step of storing a license-specifying information for specifying said license required by said user in using said content, encrypted data of said content and key information required for decrypting said encrypted data of said content;
a license storage step of storing said license including a content-specifying information for specifying said content, the use of which is allowed by the license;
a judgment step of forming a judgment as to whether or not said license required by said user in using said content has been stored in said license storage means
a decryption step, at which said encrypted data of said content is decrypted on condition that an outcome of said judgment formed at said judgment means indicates that said license required by said user in using said content has been stored in said license storage means.

11. A program according to claim 10, said program or a portion of said program encrypted.

12. A license server for issuing a license for allowing use of a content, said license server comprising:

license storage means for storing said license including:

a content-specifying information for specifying said content, use of which is allowed by said license; and
a terminal-identification information for identifying an information-processing apparatus;
receiving means for receiving a license request including a license-identification information for identifying said license from

said information-processing apparatus;
extraction means for extracting said license identified by said license-identification information included in said license request from said license storage means;
processing means for adding said terminal-identification information to said license extracted by said extraction means;
signature means for putting a signature on said license including said terminal-identification information added by said processing means by using a secret key of said license server; and
transmission means for transmitting said license with said signature put thereon by said signature means to said information-processing apparatus, from which said license request was received.

13. An information-processing method for issuing a license for allowing use of a content, said information-processing method comprising:

a license storage step of storing said license including:

a content-specifying information for specifying said content, use of which is allowed by said license; and
a terminal-identification information for identifying an information-processing apparatus;
a receiving step of receiving a license request including a license-identification information for identifying said license from said information-processing apparatus;
an extraction step of extracting said license stored in said license storage means and identified by said license-identification information included in said license request;
a processing step of adding said terminal-identification information to said license extracted at said extraction step;
a signature step of putting a signature on said license including said terminal-identification information added at said processing step by using a secret key of a license server used in said information-processing method; and
a transmission step of transmitting said license with said signature put thereon at said signature step to said information-processing apparatus, from which said license request was received.

14. A program to be executed by a computer for carrying out processing of issuing a license for allowing use of a content, said program comprising:

a license storage step of storing said license including:

a content-specifying information for specifying said content, use of which is allowed by said license; and
a terminal-identification information for identifying an information-processing apparatus;
a receiving step of receiving a license request including a license-identification information for identifying said license from said information-processing apparatus;
an extraction step of extracting said license stored in said license storage means and identified by said license-identification information included in said license request;
a processing step of adding said terminal-identification information to said license extracted at said extraction step;
a signature step of putting a signature on said license including said terminal-identification information added at said processing step by using a secret key of a license server used in said information-processing method; and
a transmission step of transmitting said license with said signature put thereon at said signature step to said information-processing apparatus, from which said license request was received.

FIG. 1

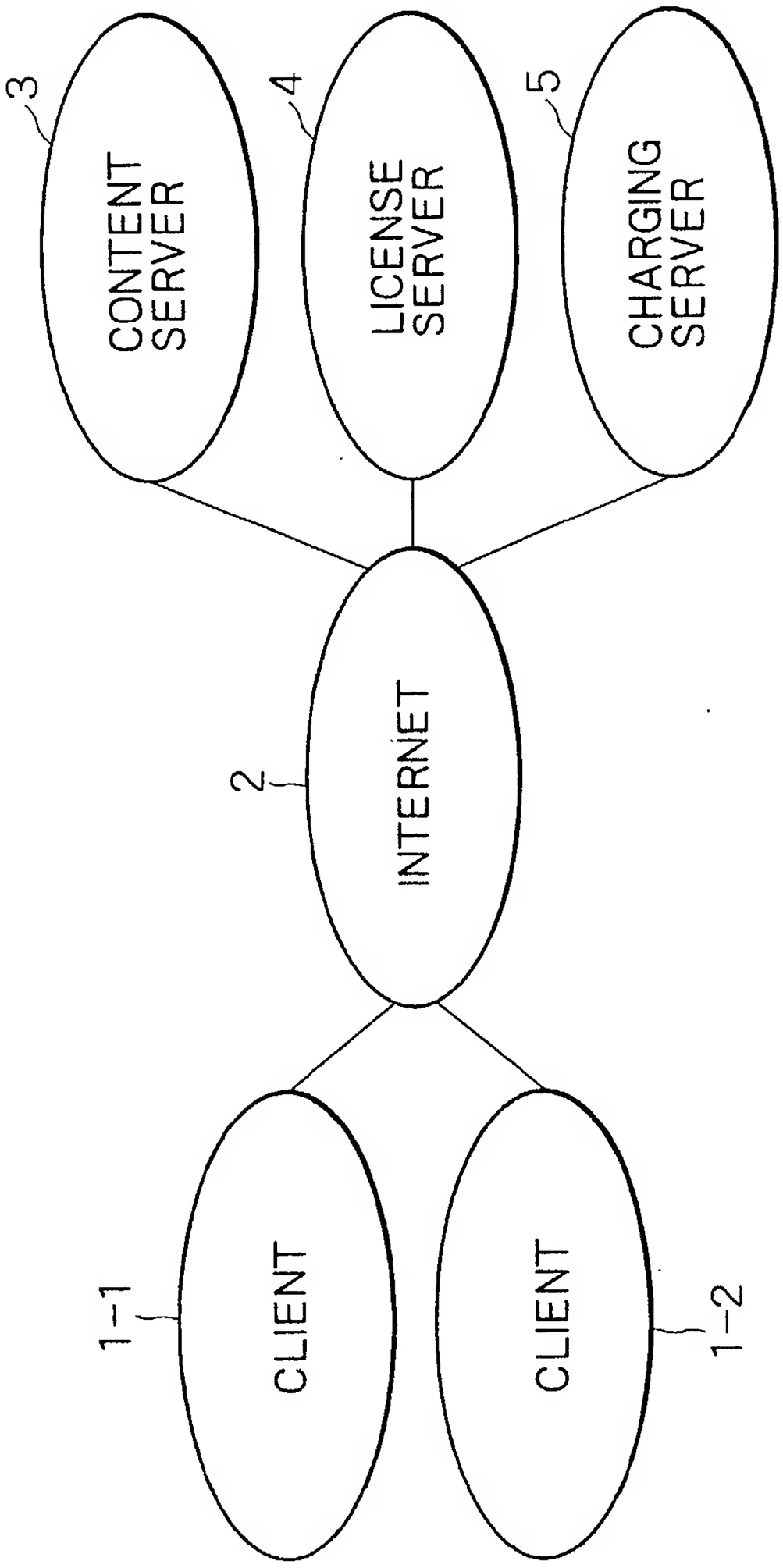

CONTENT SERVER
3
LICENSE SERVER
4
CHARGING SERVER
5
INTERNET
2
CLIENT
1-1
CLIENT
1-2

FIG. 2

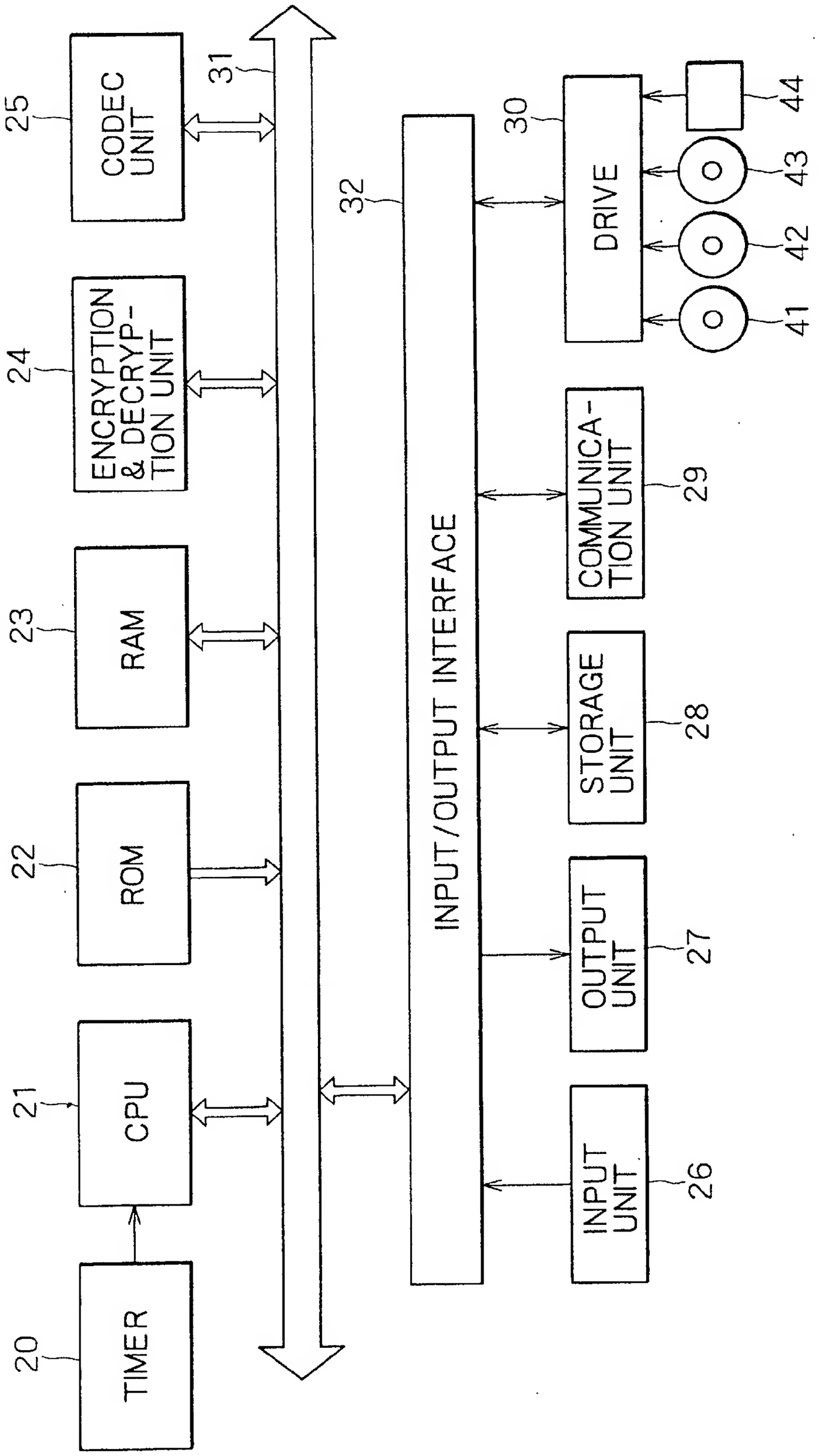

CPU
21
ROM
22
RAM
23
ENCRYPTION & DECRYPTION UNIT
24
CODEC UNIT
25
INPUT UNIT
26
OUTPUT UNIT
27
STORAGE UNIT
28
COMMUNICATION UNIT
29
DRIVE
30
31
INPUT/OUTPUT INTERFACE
32
TIMER
20
41
42
43
44

FIG. 3

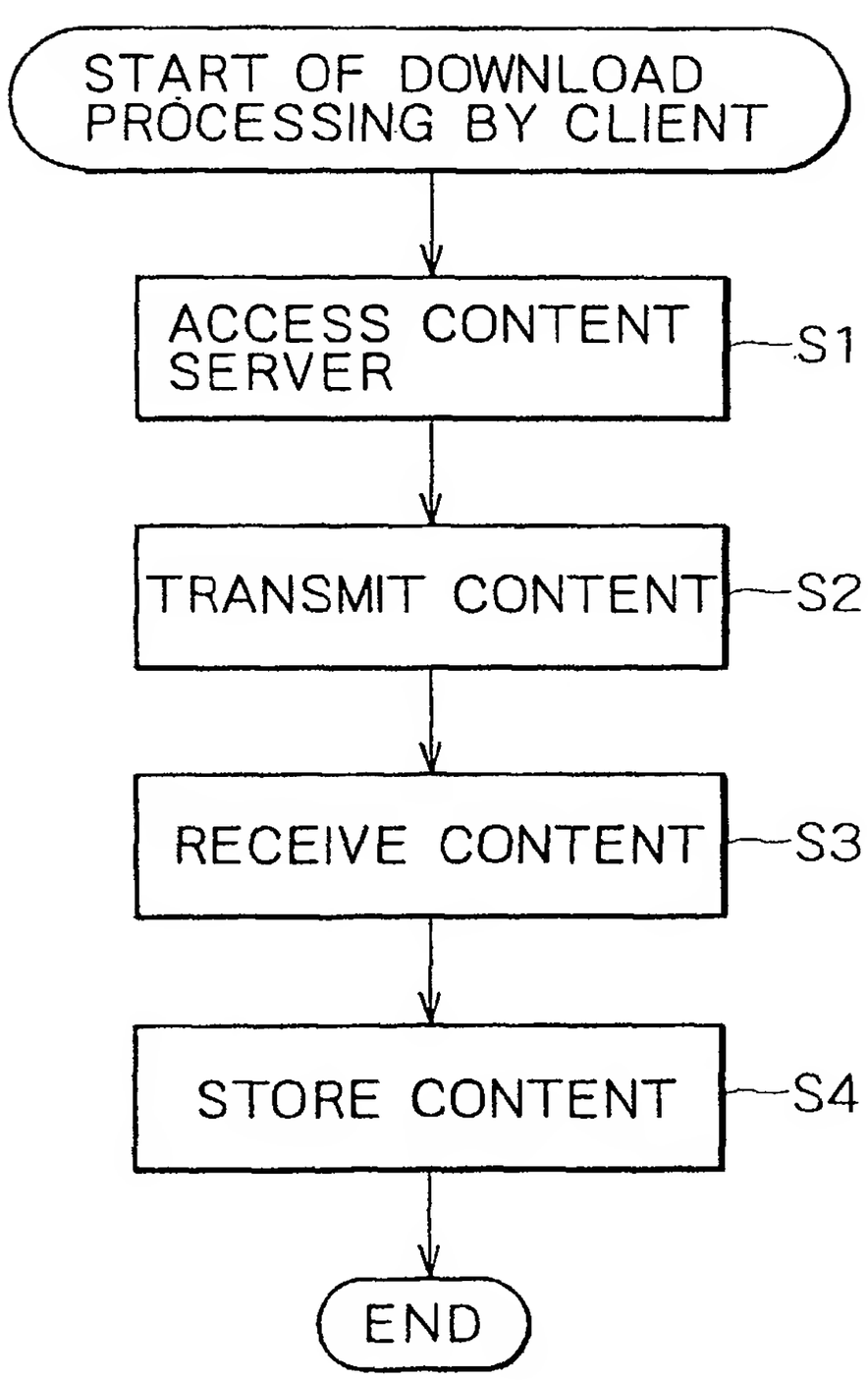

START OF DOWNLOAD PROCESSING BY CLIENT
ACCESS CONTENT SERVER
S1
TRANSMIT CONTENT
S2
RECEIVE CONTENT
S3
STORE CONTENT
S4
END

FIG. 4

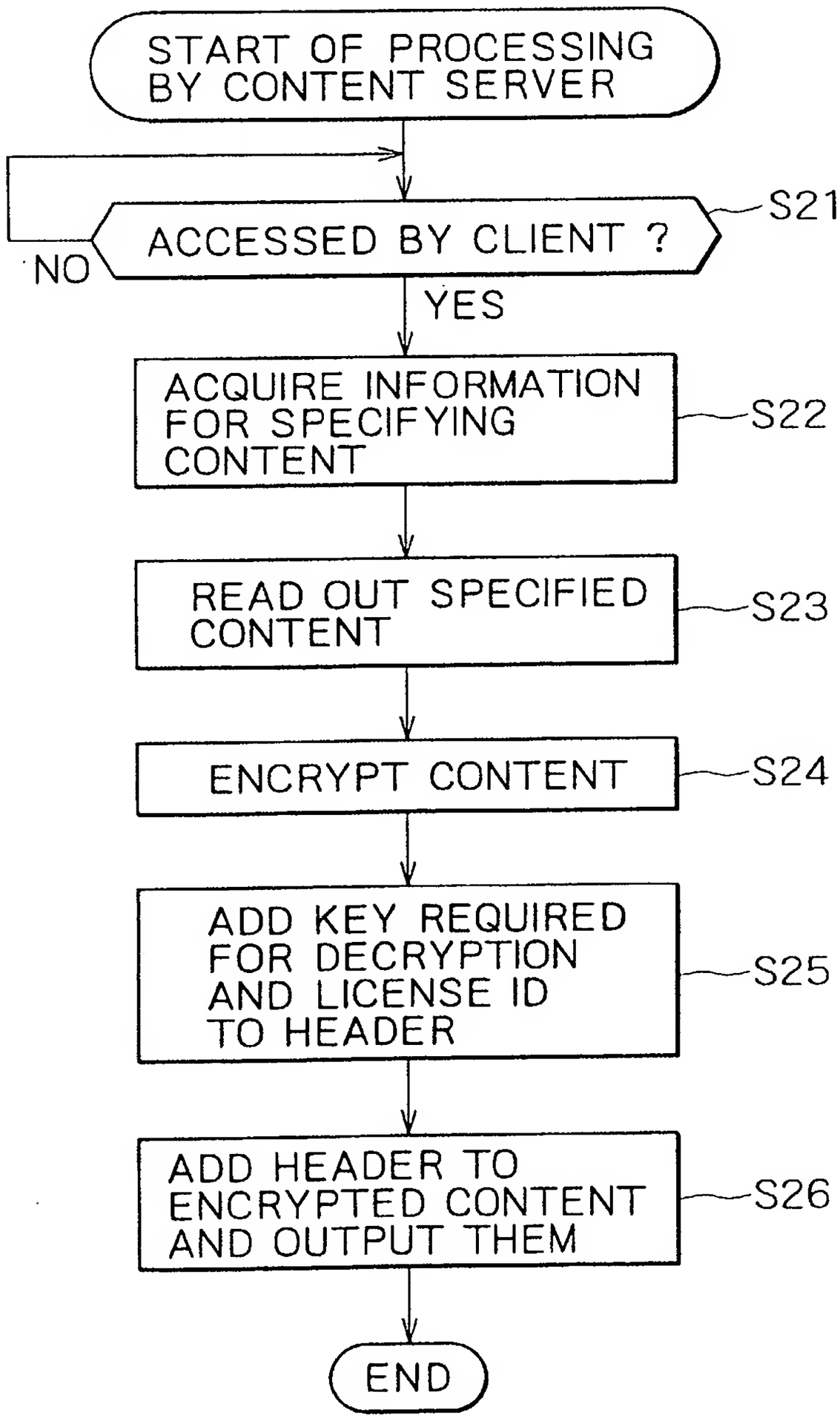

START OF PROCESSING BY CONTENT SERVER
ACCESSED BY CLIENT ?
S21
NO
YES
ACQUIRE INFORMATION FOR SPECIFYING CONTENT
S22
READ OUT SPECIFIED CONTENT
S23
ENCRYPT CONTENT
S24
ADD KEY REQUIRED FOR DECRYPTION AND LICENSE ID TO HEADER
S25
ADD HEADER TO ENCRYPTED CONTENT AND OUTPUT THEM
S26
END

FIG.5

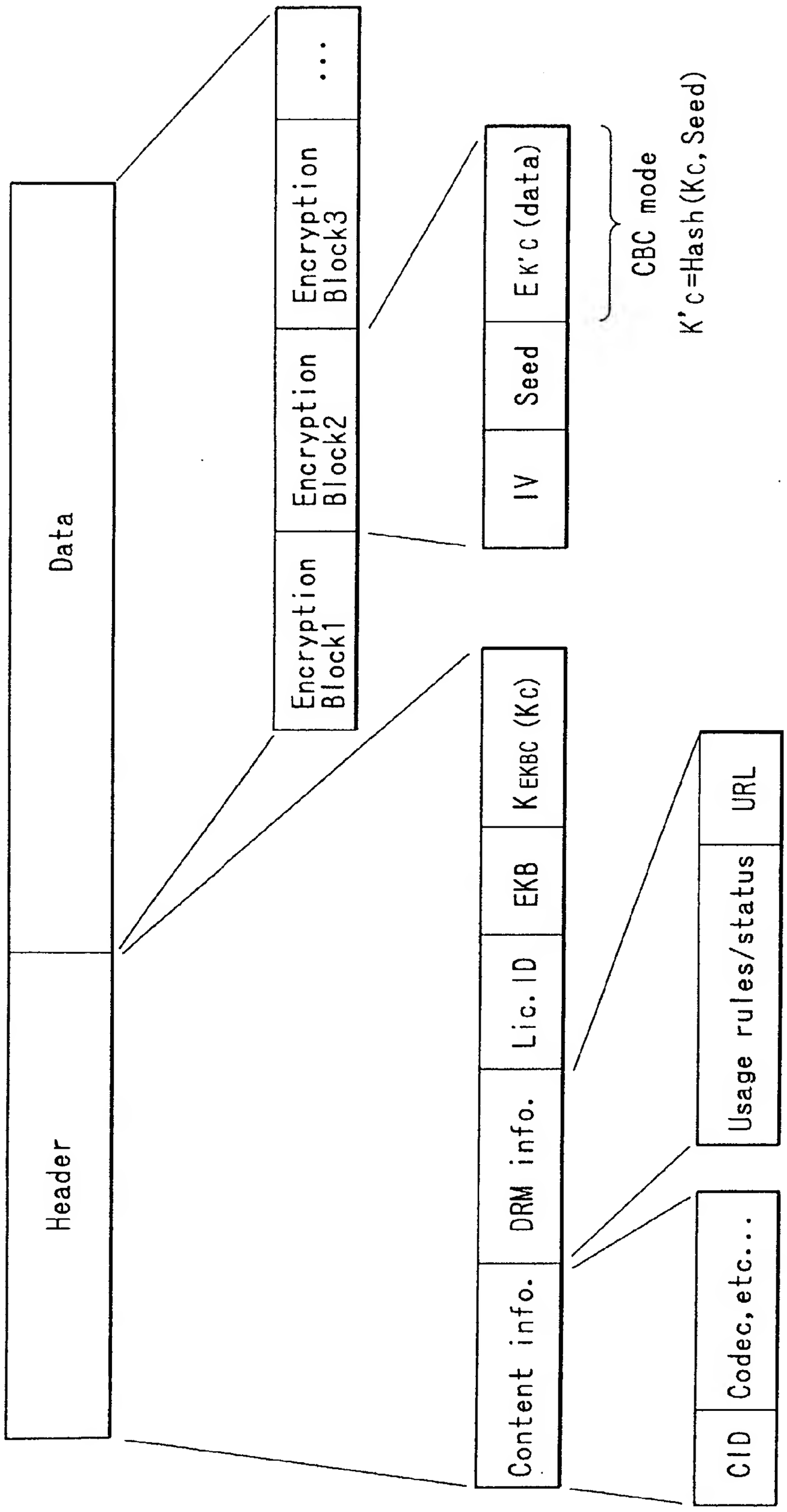

Header
Data
Encryption Block1
Encryption Block2
Encryption Block3
...
IV
Seed
E K'c (data)
CBC mode
K'c=Hash (Kc, Seed)
DRM info.
Lic. ID
EKB
K EKBC (Kc)
Content info.
CID
Codec, etc...
Usage rules/status
URL

FIG. 6

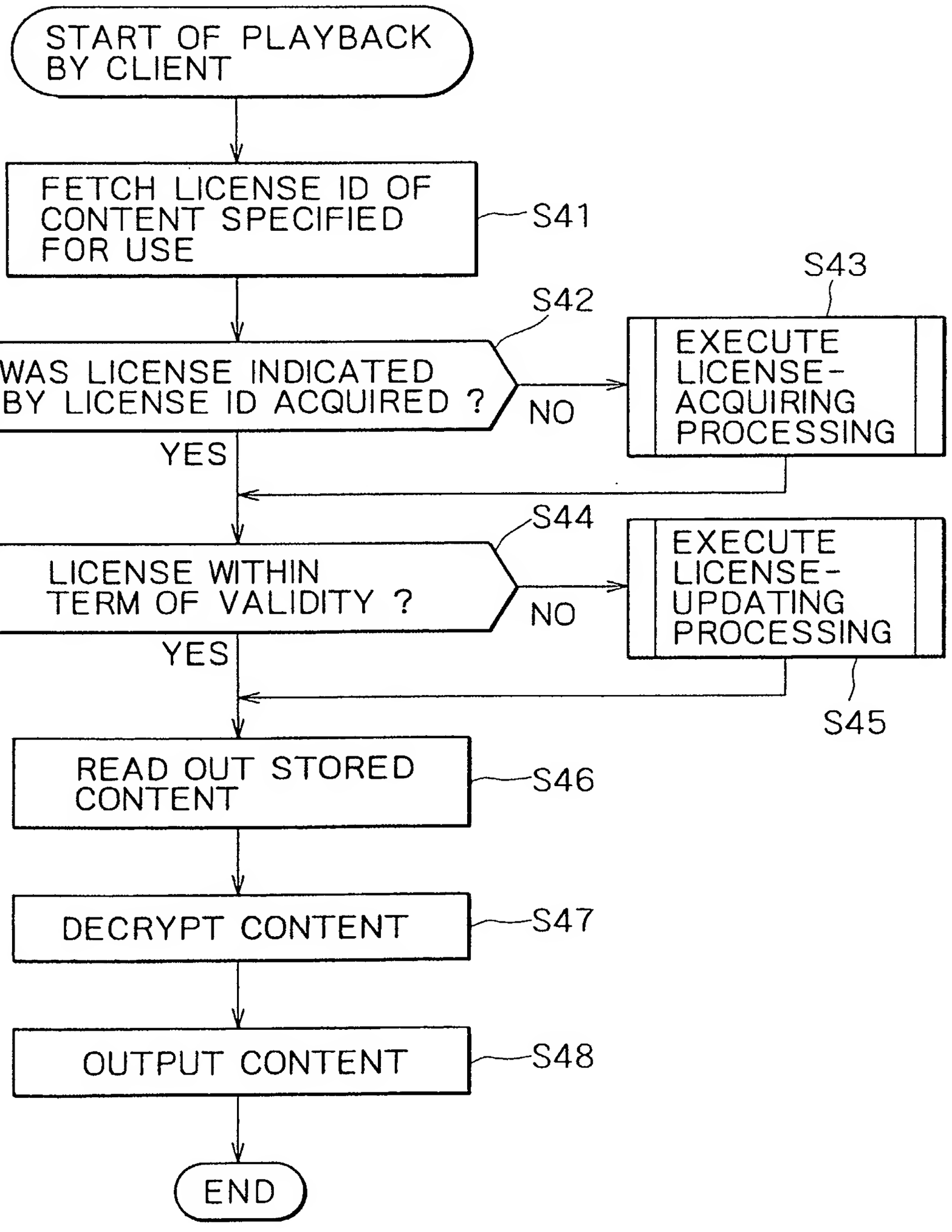

START OF PLAYBACK BY CLIENT
FETCH LICENSE ID OF CONTENT SPECIFIED FOR USE
S41
WAS LICENSE INDICATED BY LICENSE ID ACQUIRED ?
S42
NO
YES
EXECUTE LICENSE-ACQUIRING PROCESSING
S43
LICENSE WITHIN TERM OF VALIDITY ?
S44
NO
YES
EXECUTE LICENSE-UPDATING PROCESSING
S45
READ OUT STORED CONTENT
S46
DECRYPT CONTENT
S47
OUTPUT CONTENT
S48
END

FIG. 7

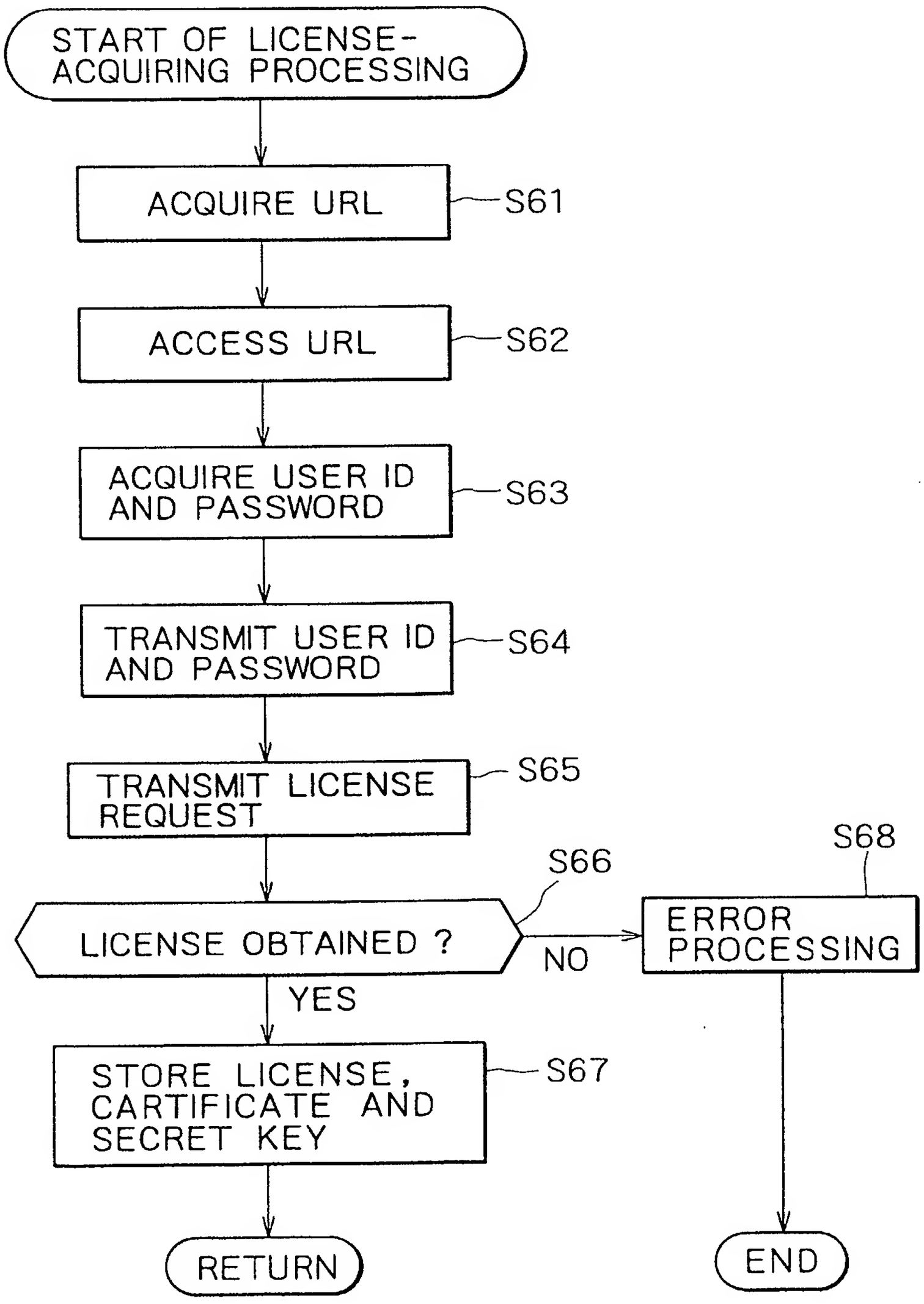

START OF LICENSE-ACQUIRING PROCESSING
ACQUIRE URL
S61
ACCESS URL
S62
ACQUIRE USER ID AND PASSWORD
S63
TRANSMIT USER ID AND PASSWORD
S64
TRANSMIT LICENSE REQUEST
S65
S66
LICENSE OBTAINED ?
NO
S68
ERROR PROCESSING
YES
STORE LICENSE, CARTIFICATE AND SECRET KEY
S67
RETURN
END

FIG.8

LICENSE ID
CREATION DATE AND TIME
VALIDITY TERM
USAGE CONDITION
LEAF ID
DIGITAL SIGNATURE

FIG. 9

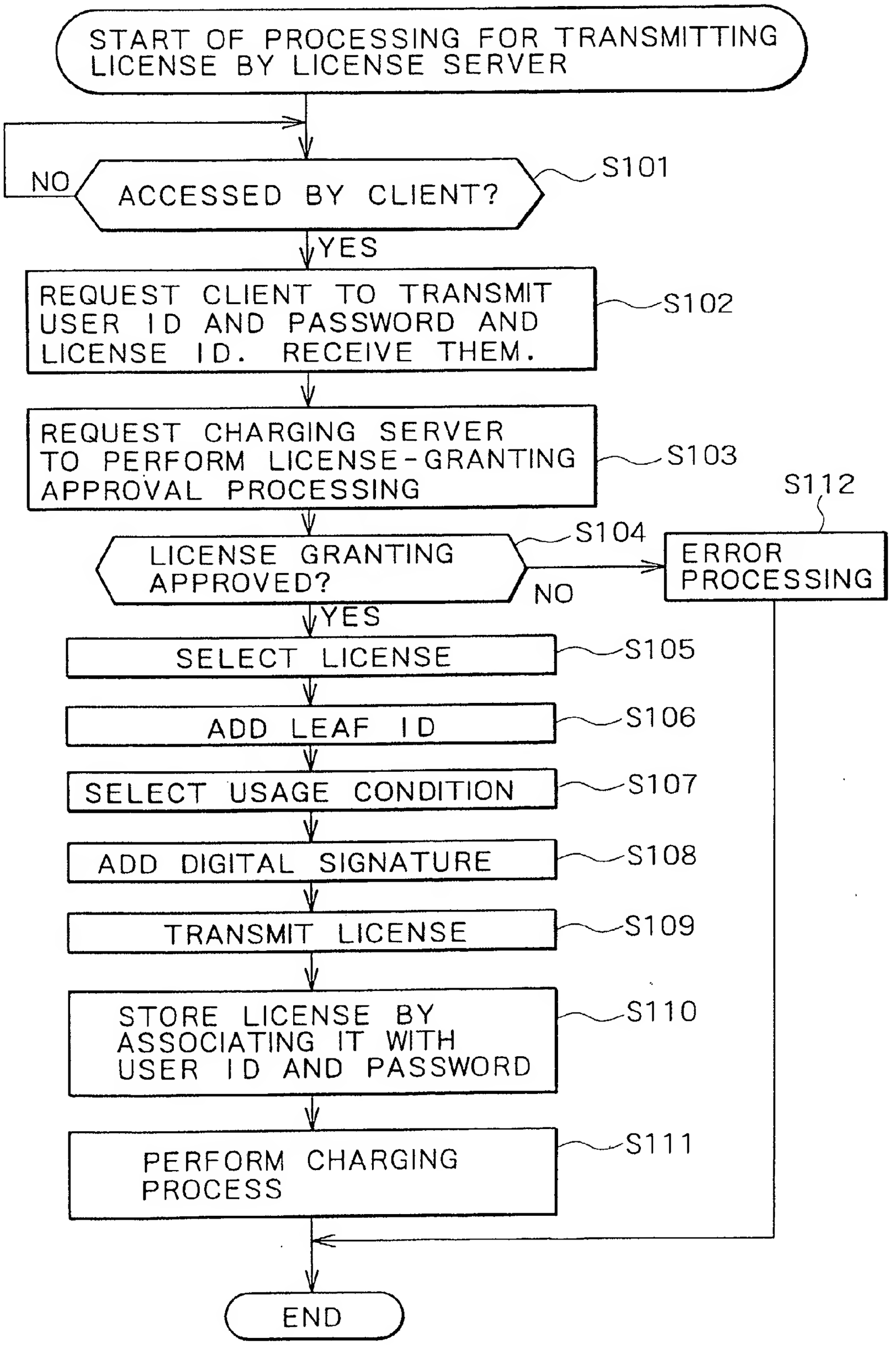

START OF PROCESSING FOR TRANSMITTING LICENSE BY LICENSE SERVER
NO
ACCESSED BY CLIENT?
S101
YES
REQUEST CLIENT TO TRANSMIT USER ID AND PASSWORD AND LICENSE ID. RECEIVE THEM.
S102
REQUEST CHARGING SERVER TO PERFORM LICENSE-GRANTING APPROVAL PROCESSING
S103
LICENSE GRANTING APPROVED?
S104
NO
S112
ERROR PROCESSING
YES
SELECT LICENSE
S105
ADD LEAF ID
S106
SELECT USAGE CONDITION
S107
ADD DIGITAL SIGNATURE
S108
TRANSMIT LICENSE
S109
STORE LICENSE BY ASSOCIATING IT WITH USER ID AND PASSWORD
S110
PERFORM CHARGING PROCESS
S111
END

FIG. 10

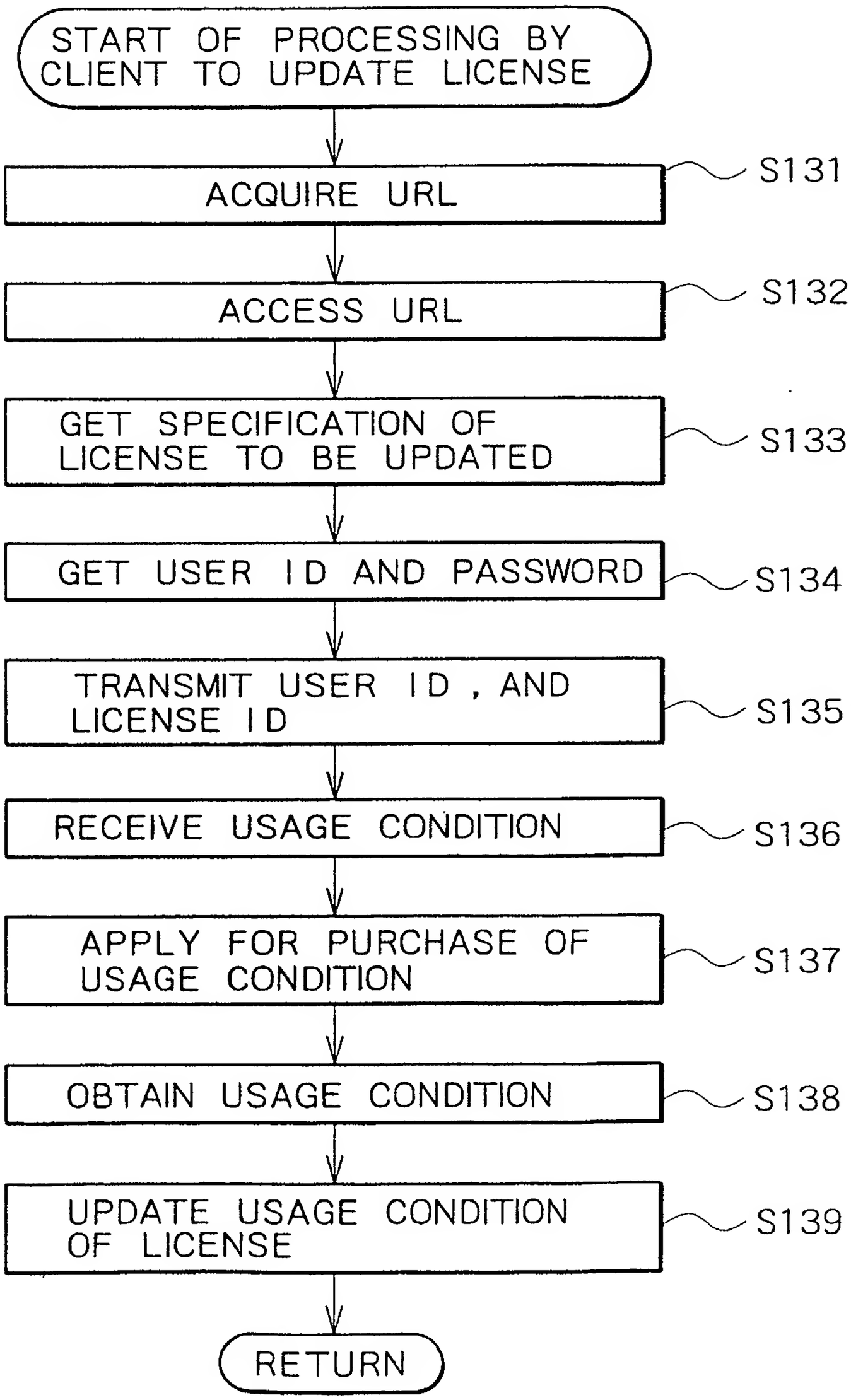

START OF PROCESSING BY CLIENT TO UPDATE LICENSE
ACQUIRE URL
S131
ACCESS URL
S132
GET SPECIFICATION OF LICENSE TO BE UPDATED
S133
GET USER ID AND PASSWORD
S134
TRANSMIT USER ID , AND LICENSE ID
S135
RECEIVE USAGE CONDITION
S136
APPLY FOR PURCHASE OF USAGE CONDITION
S137
OBTAIN USAGE CONDITION
S138
UPDATE USAGE CONDITION OF LICENSE
S139
RETURN

FIG. 11

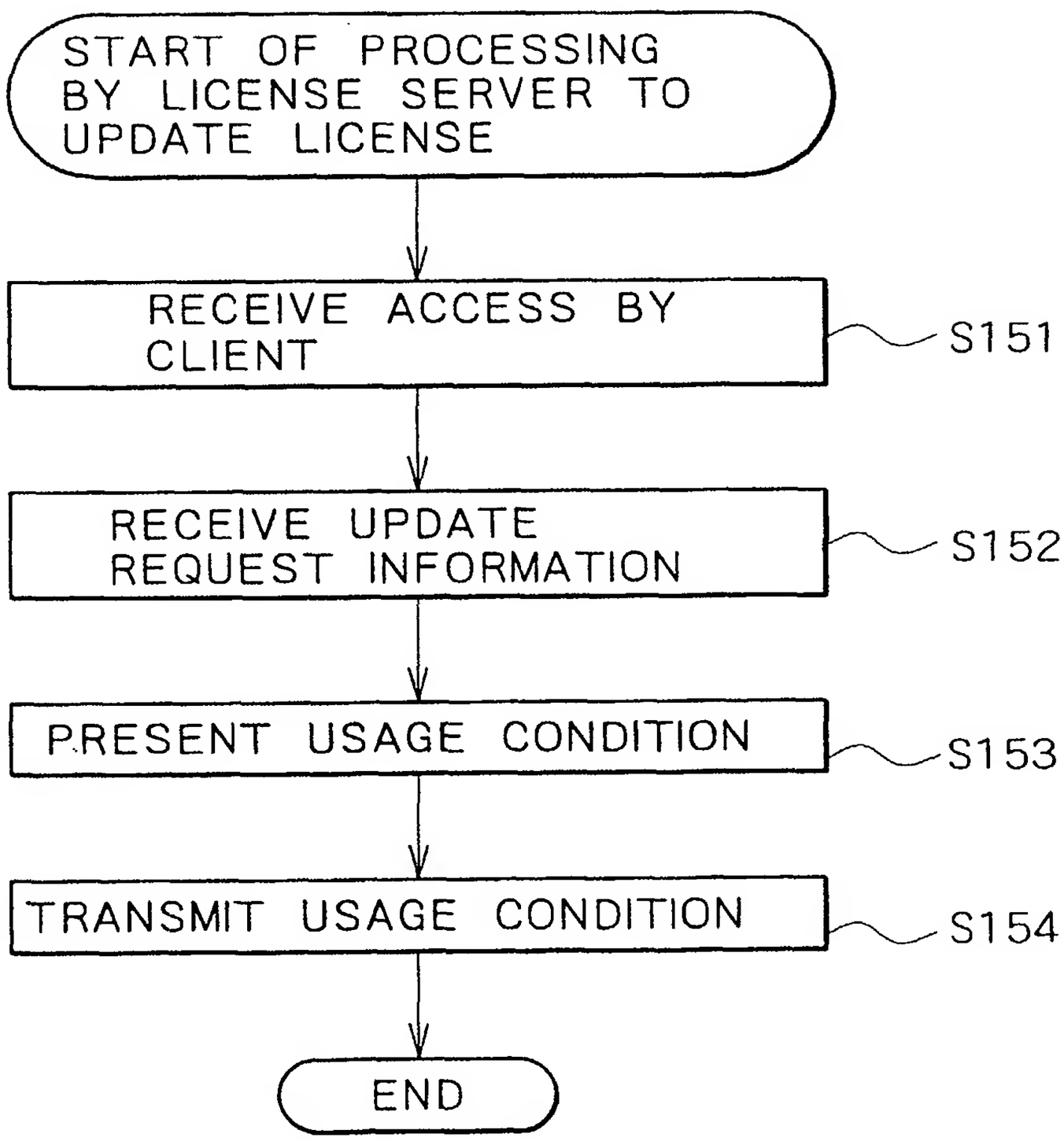

START OF PROCESSING BY LICENSE SERVER TO UPDATE LICENSE
RECEIVE ACCESS BY CLIENT
S151
RECEIVE UPDATE REQUEST INFORMATION
S152
PRESENT USAGE CONDITION
S153
TRANSMIT USAGE CONDITION
S154
END

FIG. 12

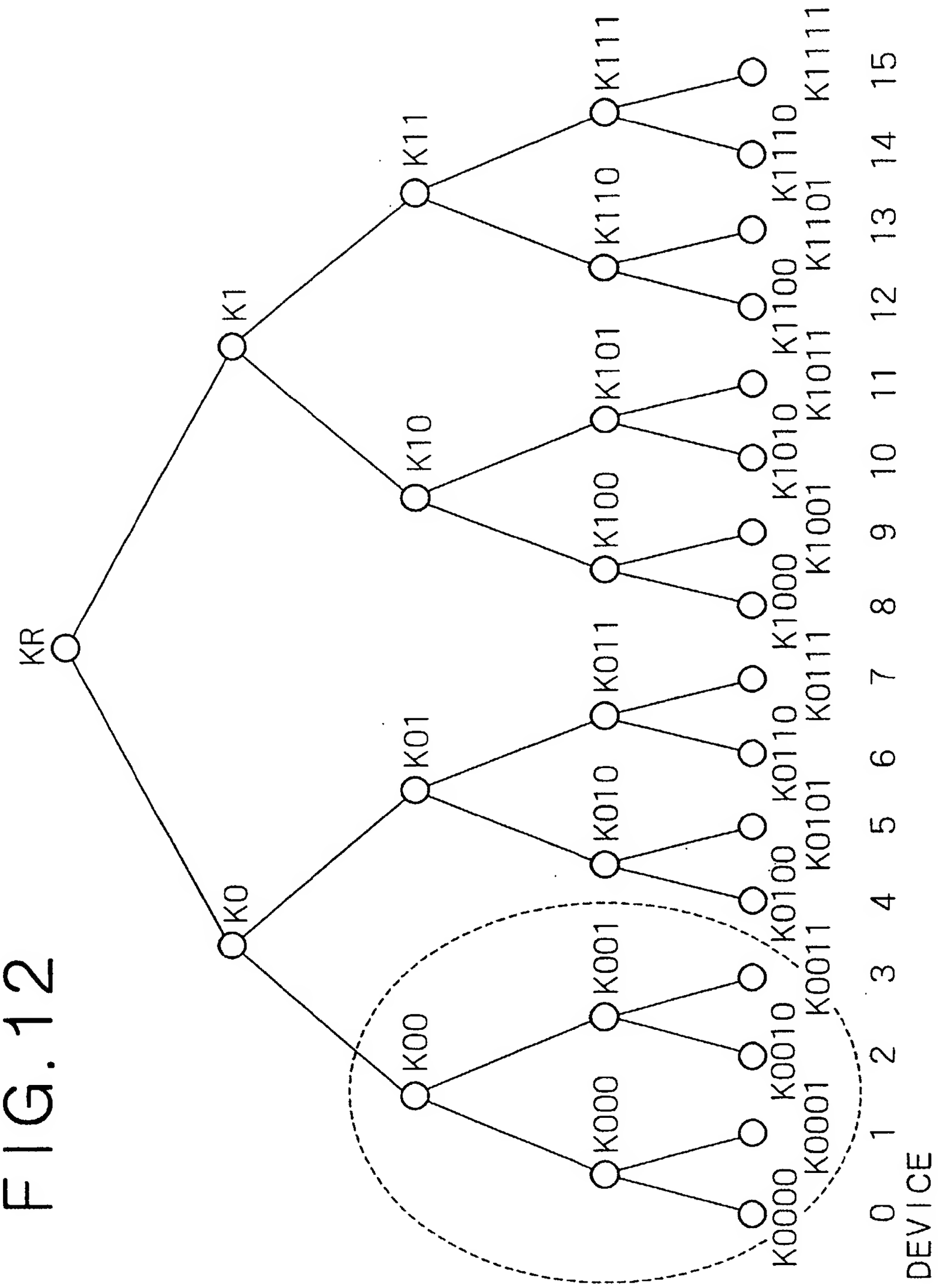

KR
K0
K1
K00
K01
K10
K11
K000
K001
K010
K011
K100
K101
K110
K111
K0000
K0001
K0010
K0011
K0100
K0101
K0110
K0111
K1000
K1001
K1010
K1011
K1100
K1101
K1110
K1111
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
DEVICE

FIG.13

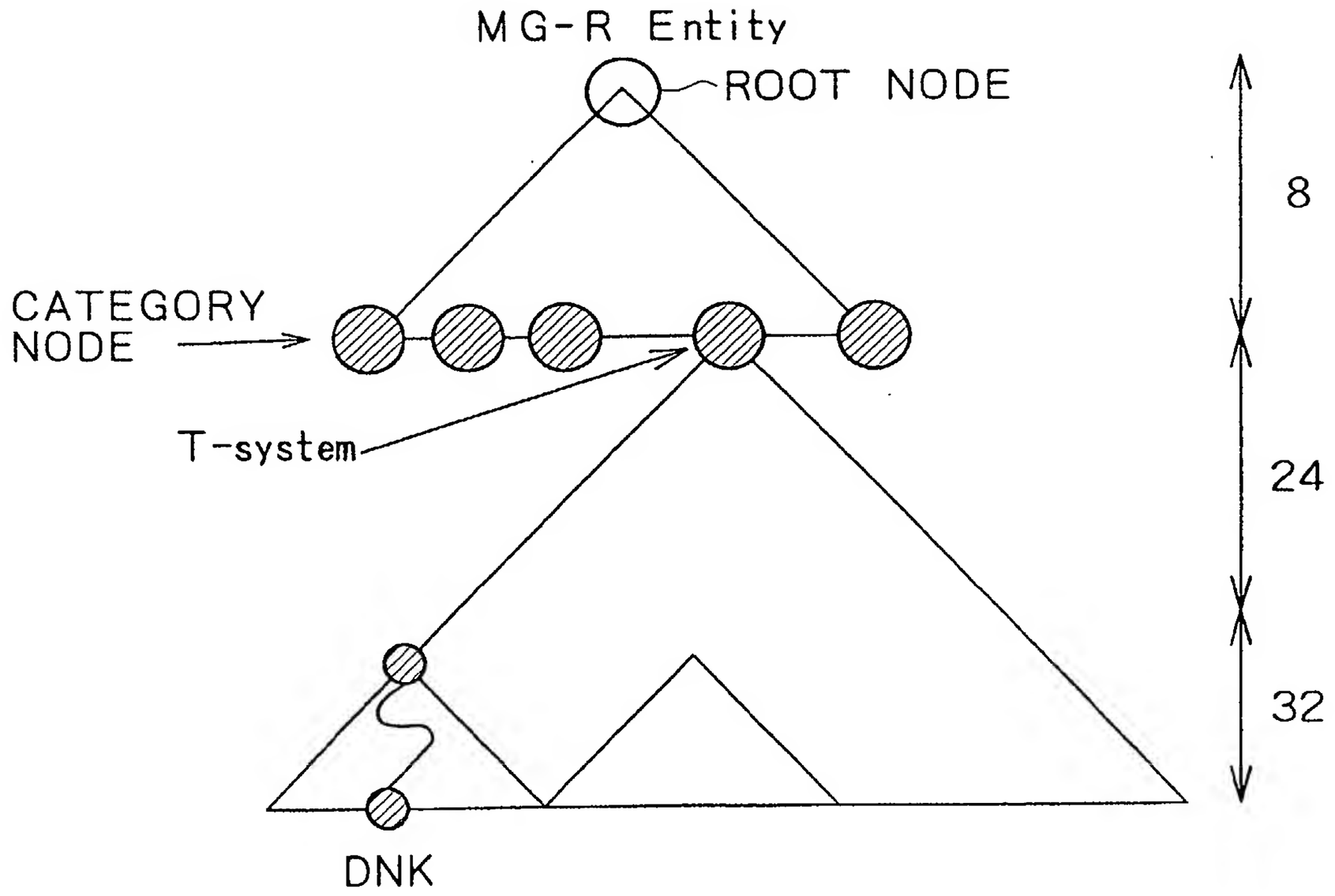


FIG. 14

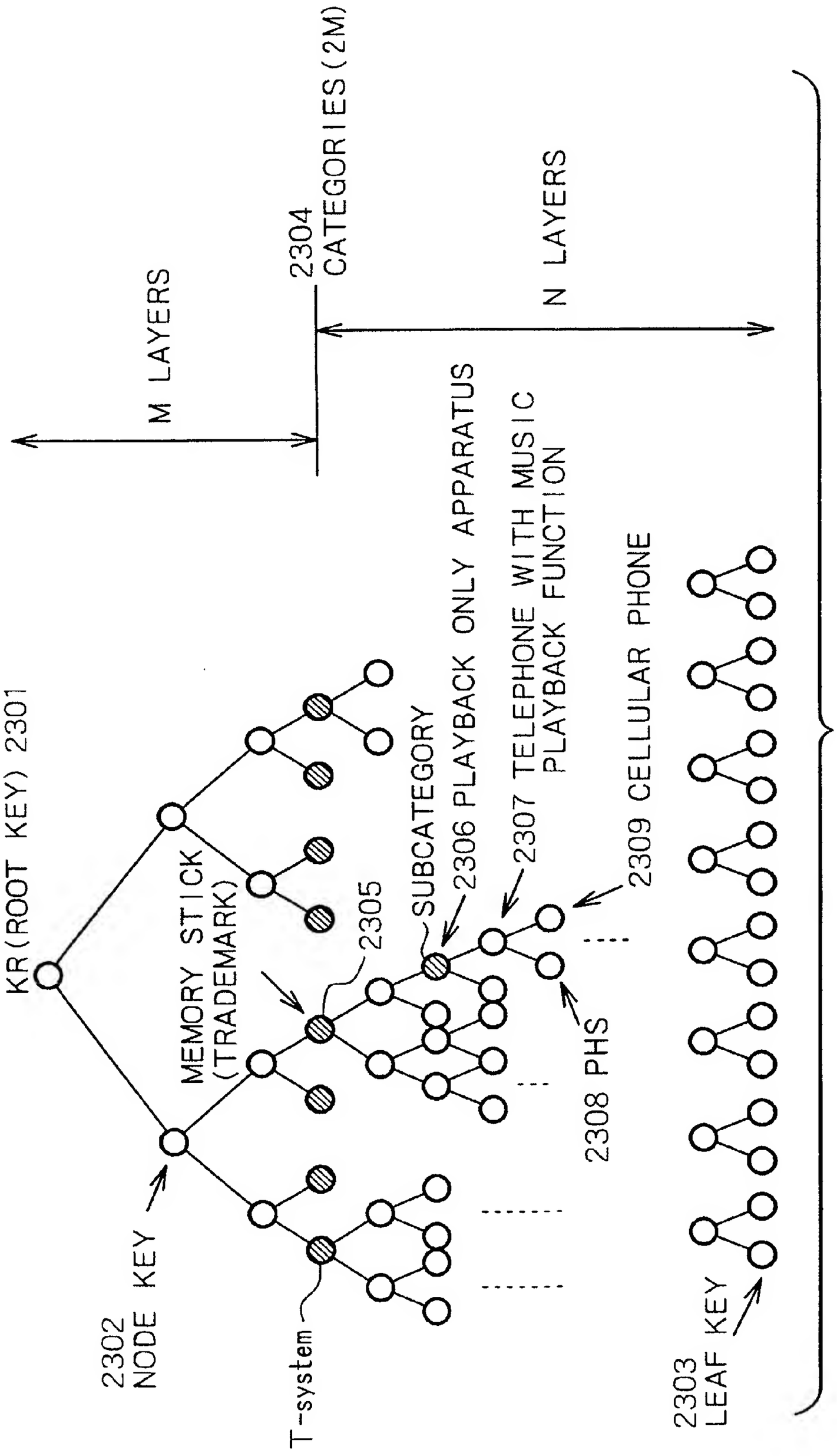

KR (ROOT KEY) 2301
2302 NODE KEY
2303 LEAF KEY
MEMORY STICK (TRADEMARK)
2304 CATEGORIES (2M)
M LAYERS
N LAYERS
2305
SUBCATEGORY
2306 PLAYBACK ONLY APPARATUS
2307 TELEPHONE WITH MUSIC PLAYBACK FUNCTION
2308 PHS
2309 CELLULAR PHONE
T-system

FIG. 15A

EKB (ENABLING KEY BLOCK)
TRANSMISSION OF NODE KEYS OF VERSION t TO DEVICES 0, 1 AND 2

VERSION:t	
INDEX	ENCRYPTION KEY
0	Enc(K(t)0, K(t)R)
00	Enc(K(t)00, K(t)0)
000	Enc(K000, K(t)00)
001	Enc(K(t)001, K(t)00)
0010	Enc(K0010, K(t)001)

FIG. 15B

EKB (ENABLING KEY BLOCK)
TRANSMIT NODE KEYS OF VERSION t TO DEVICES 0, 1 AND 2

VERSION:t	
INDEX	ENCRYPTION KEY
000	Enc(K000, K(t)00)
001	Enc(K(t)001, K(t)00)
0010	Enc(K0010, K(t)001)

FIG. 16

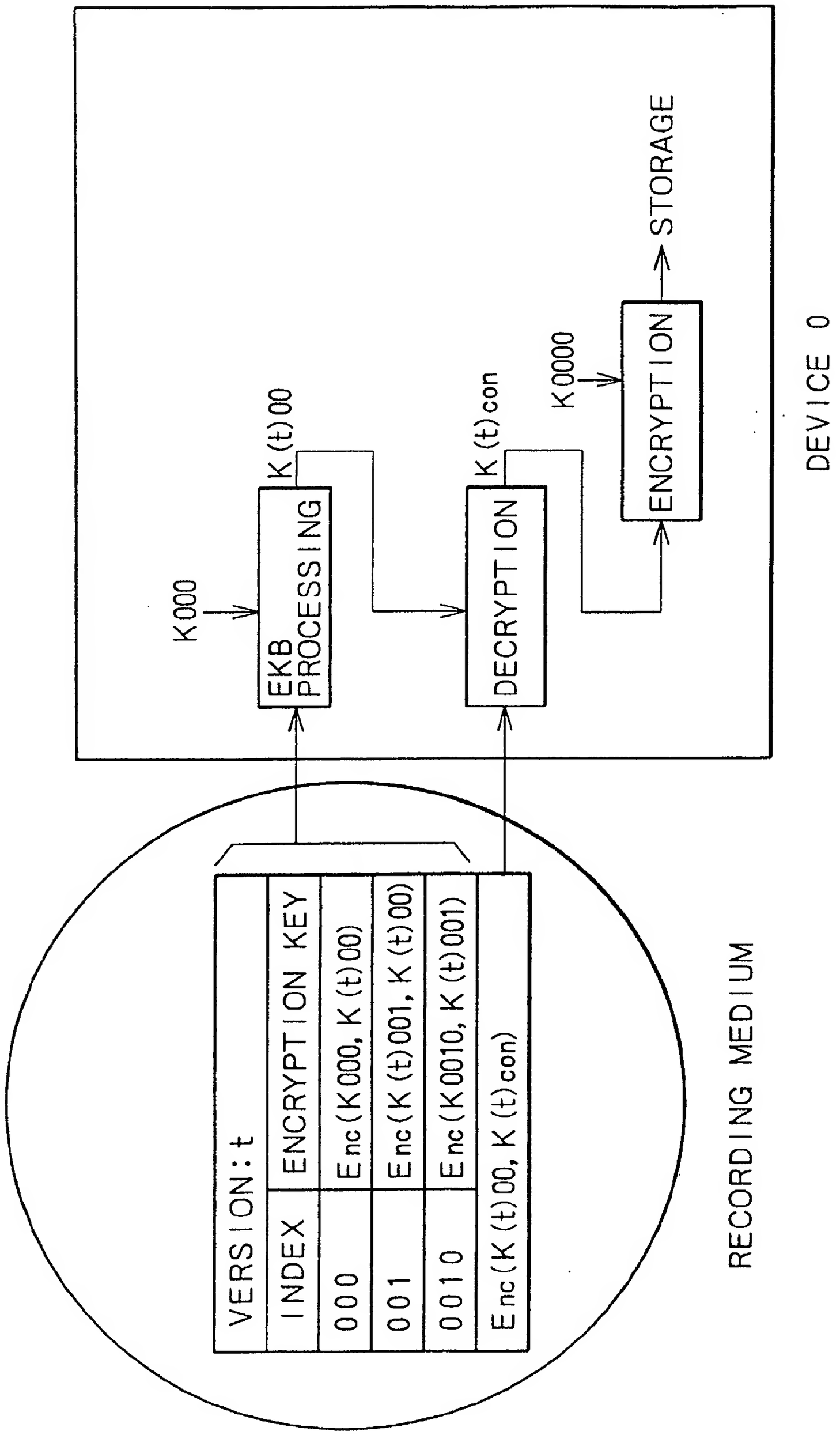

DEVICE 0
K000
EKB PROCESSING
K (t)00
K (t)con
DECRYPTION
K000
ENCRYPTION
STORAGE
RECORDING MEDIUM
VERSION: t
INDEX
ENCRYPTION KEY
000
Enc (K000, K (t)00)
001
Enc (K (t)001, K (t)00)
0010
Enc (K0010, K (t)001)
Enc (K (t)00, K (t)con)

FIG. 17

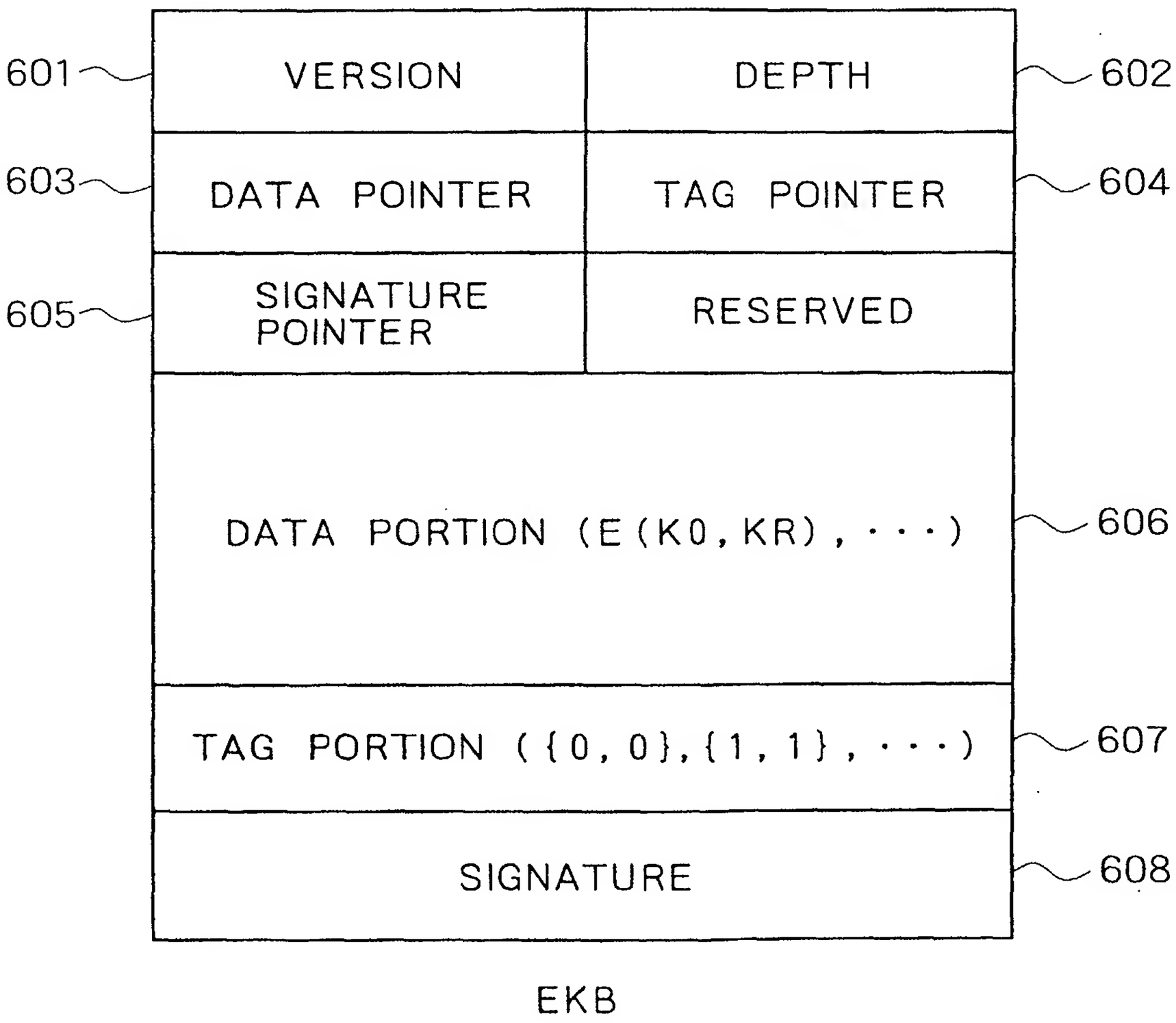

601
VERSION
DEPTH
602
603
DATA POINTER
TAG POINTER
604
605
SIGNATURE POINTER
RESERVED
DATA PORTION (E(K0, KR), ···)
606
TAG PORTION ({0, 0}, {1, 1}, ···)
607
SIGNATURE
608
EKB

FIG. 18A

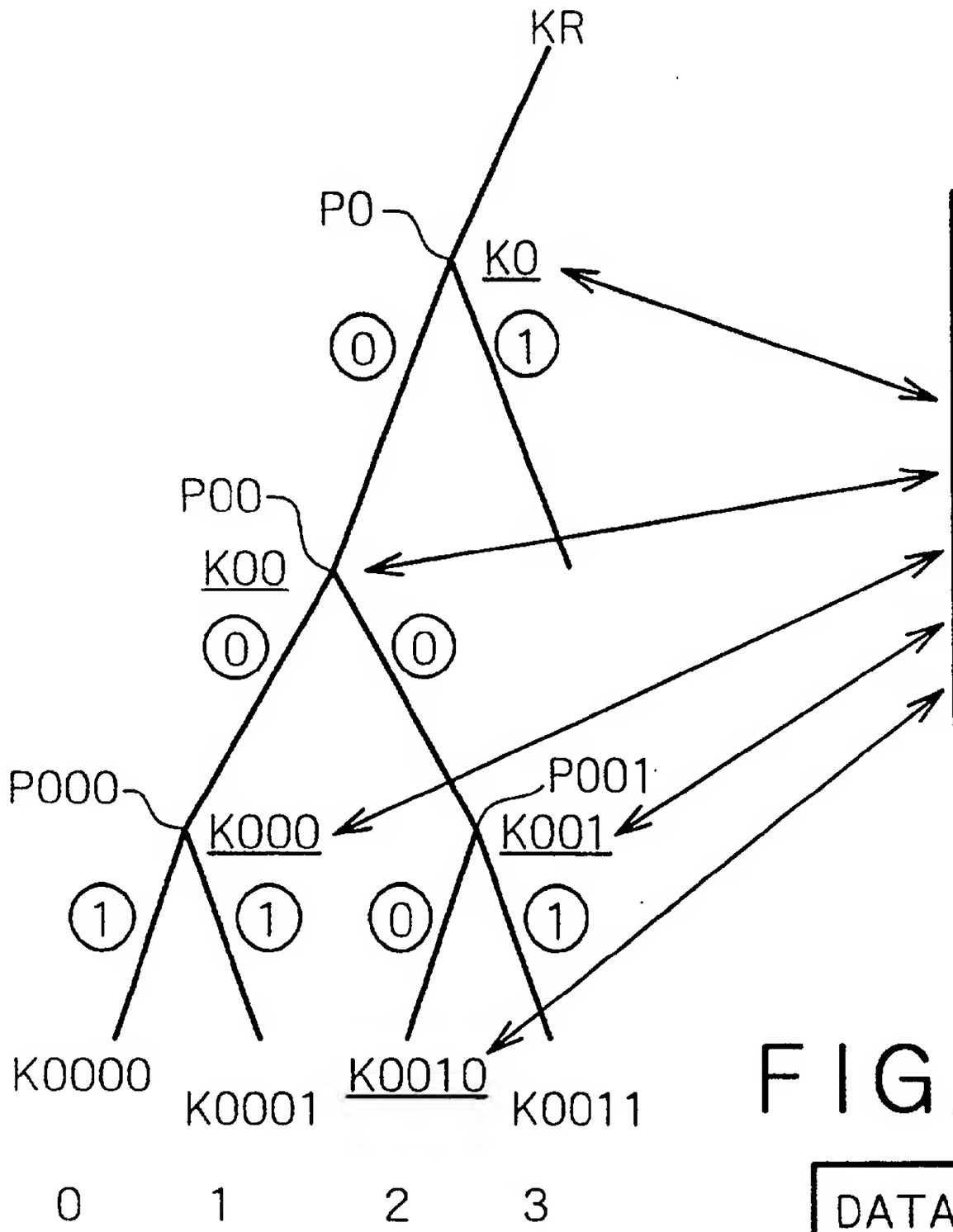


FIG. 18B

EKB (ENABLING KEY BLOCK)

TRANSMISSION OF NODE KEYS OF VERSION t TO DEVICES 0, 1 AND 2

TOP NODE ADDRESS:KR	
DATA (ENCRYPTION KEY)	TAG
Enc(K(t)0, K(t)R)	{0, 1}
Enc(K(t)00, K(t)0)	{0, 0}
Enc(K000, K(t)00)	{1, 1}
Enc(K(t)001, K(t)00)	{0, 1}
Enc(K0010, K(t)001)	{1, 1}

{L TAG, R TAG}
L TAG AND R TAG ARE EACH 0 TO INDICATE EXISTENCE OF DATA OR 1 OTHERWISE

FIG. 18C

DATA: Enc(K(t)0, K(t)R), Eno(K(t)00, K(t)0), ···
TAG: {0, 1}, {0, 0}, {1, 1} ···

EP 1 231 532 A2

FIG. 19

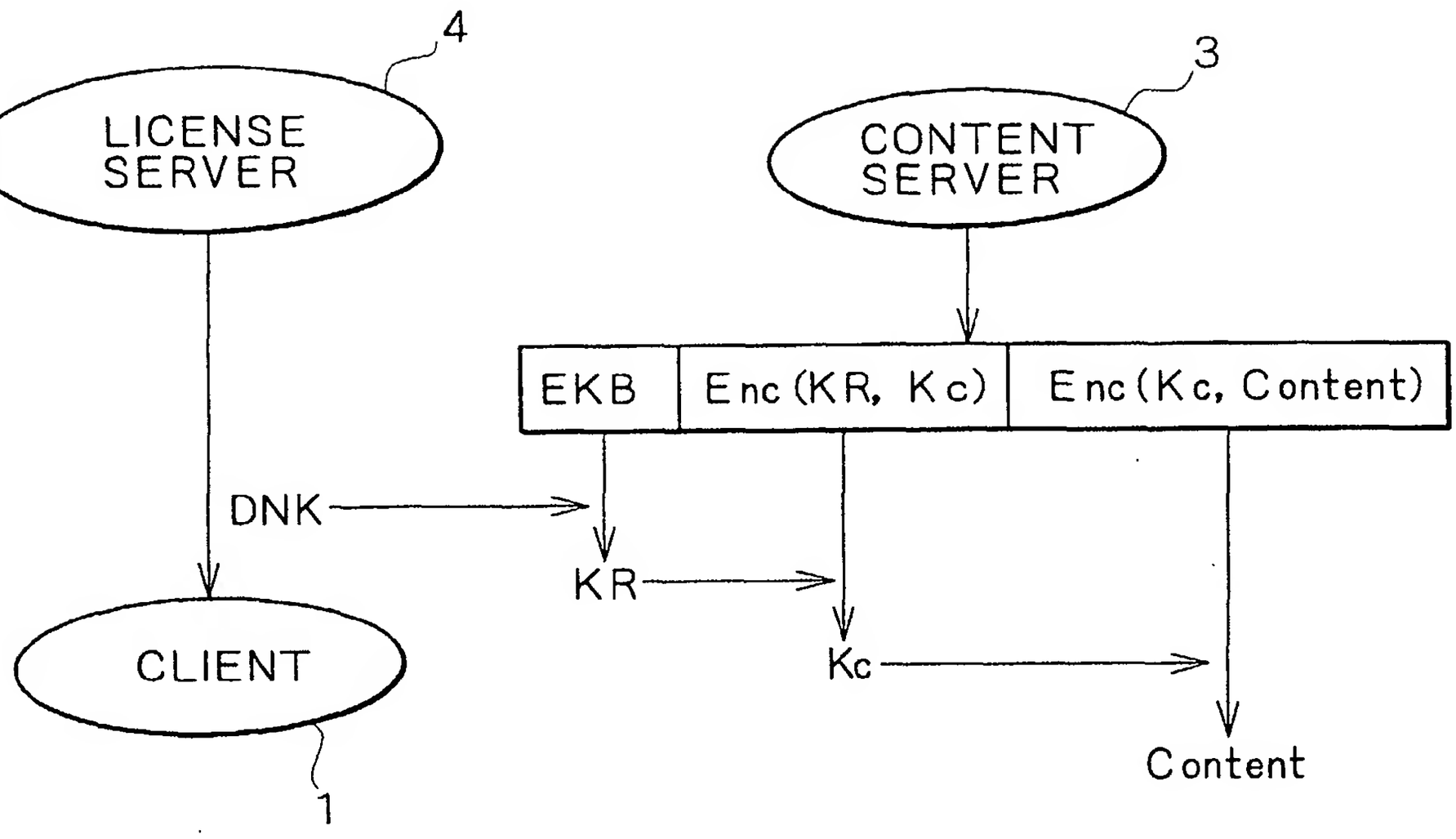

4
LICENSE SERVER
3
CONTENT SERVER
EKB
Enc(KR, Kc)
Enc(Kc, Content)
DNK
KR
Kc
CLIENT
Content
1

FIG. 20

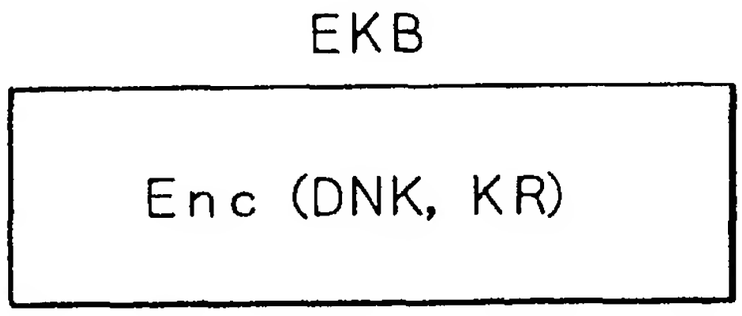

EKB
Enc (DNK, KR)

FIG.21

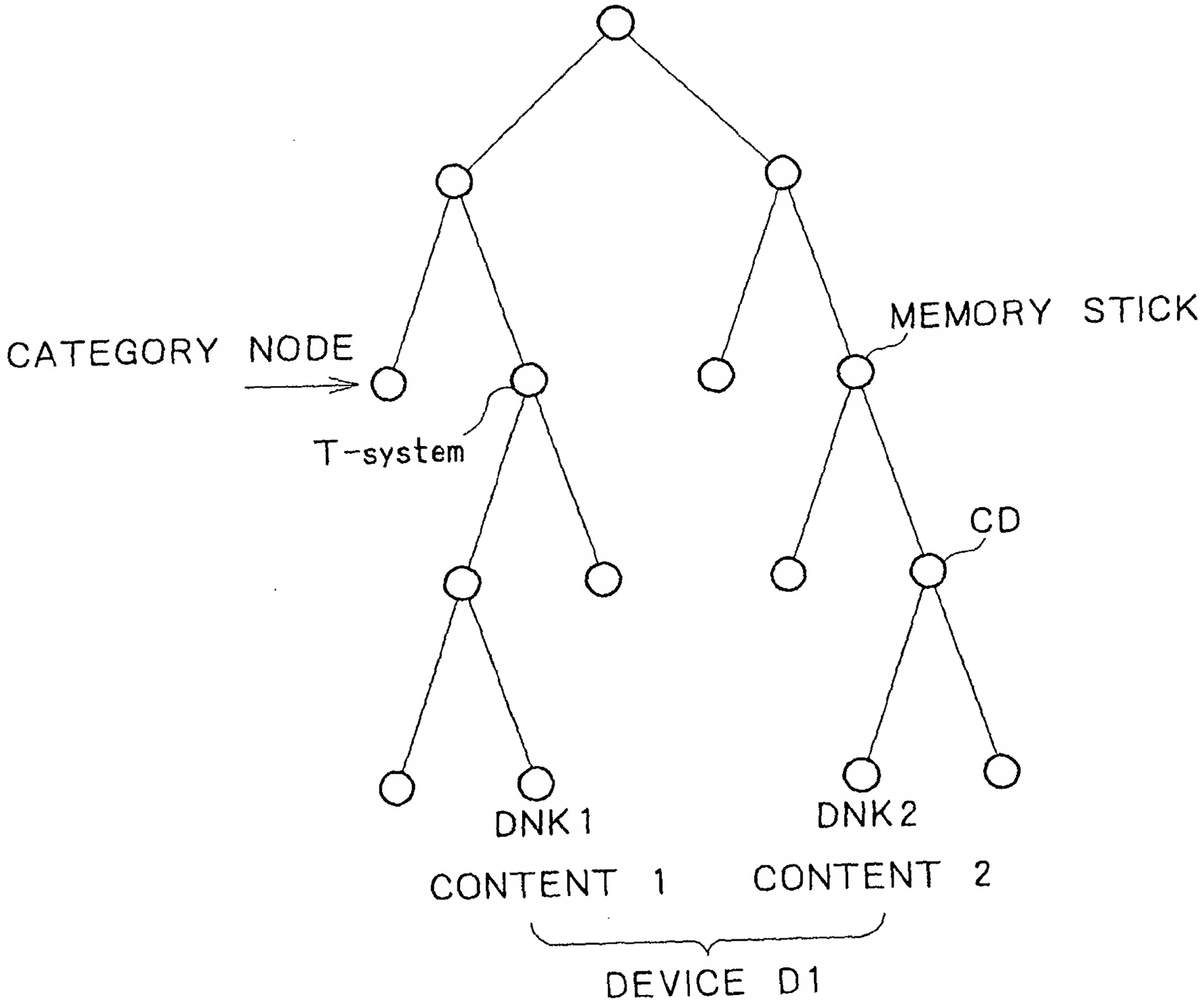

CATEGORY NODE
T-system
MEMORY STICK
CD
DNK1
DNK2
CONTENT 1
CONTENT 2
DEVICE D1

FIG. 22

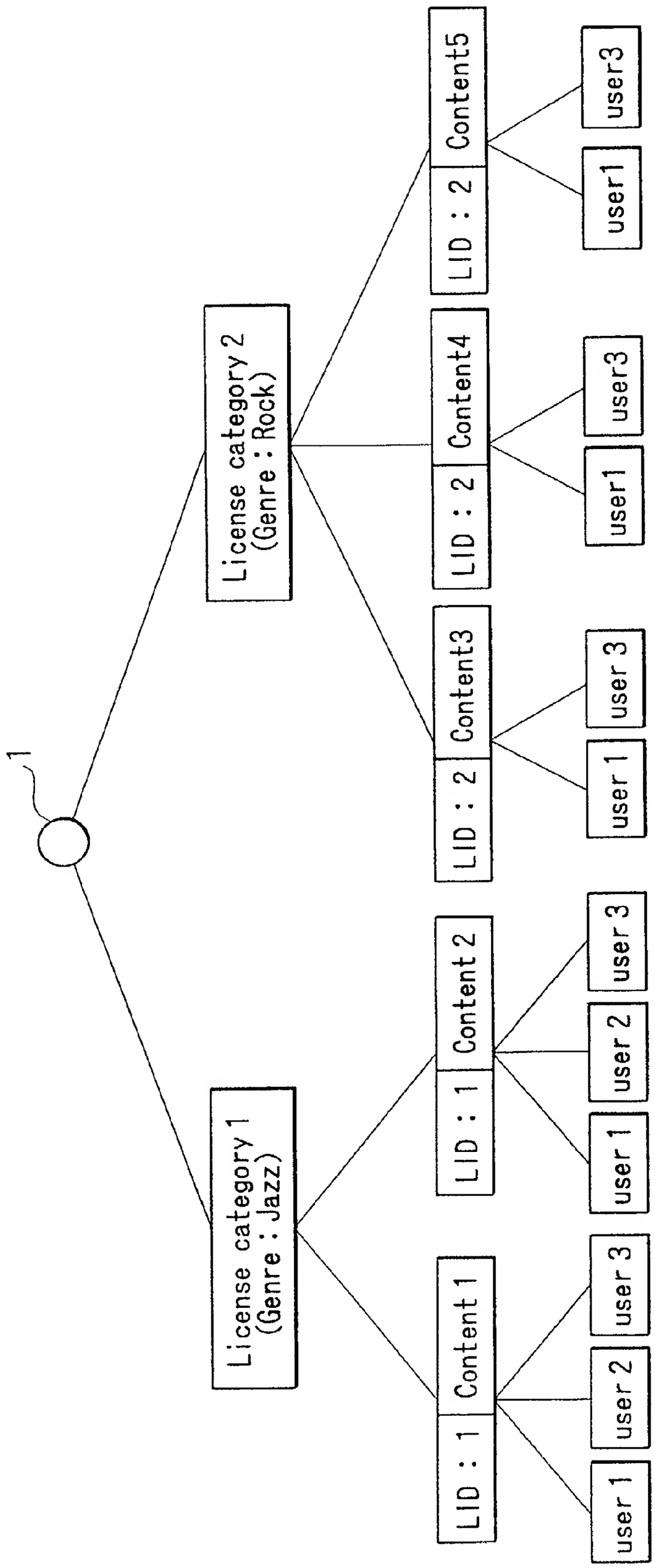

1
License category 1 (Genre : Jazz)
License category 2 (Genre : Rock)
LID : 1
Content 1
Content 2
LID : 2
Content3
Content4
Content5
user 1
user 2
user 3

FIG. 23

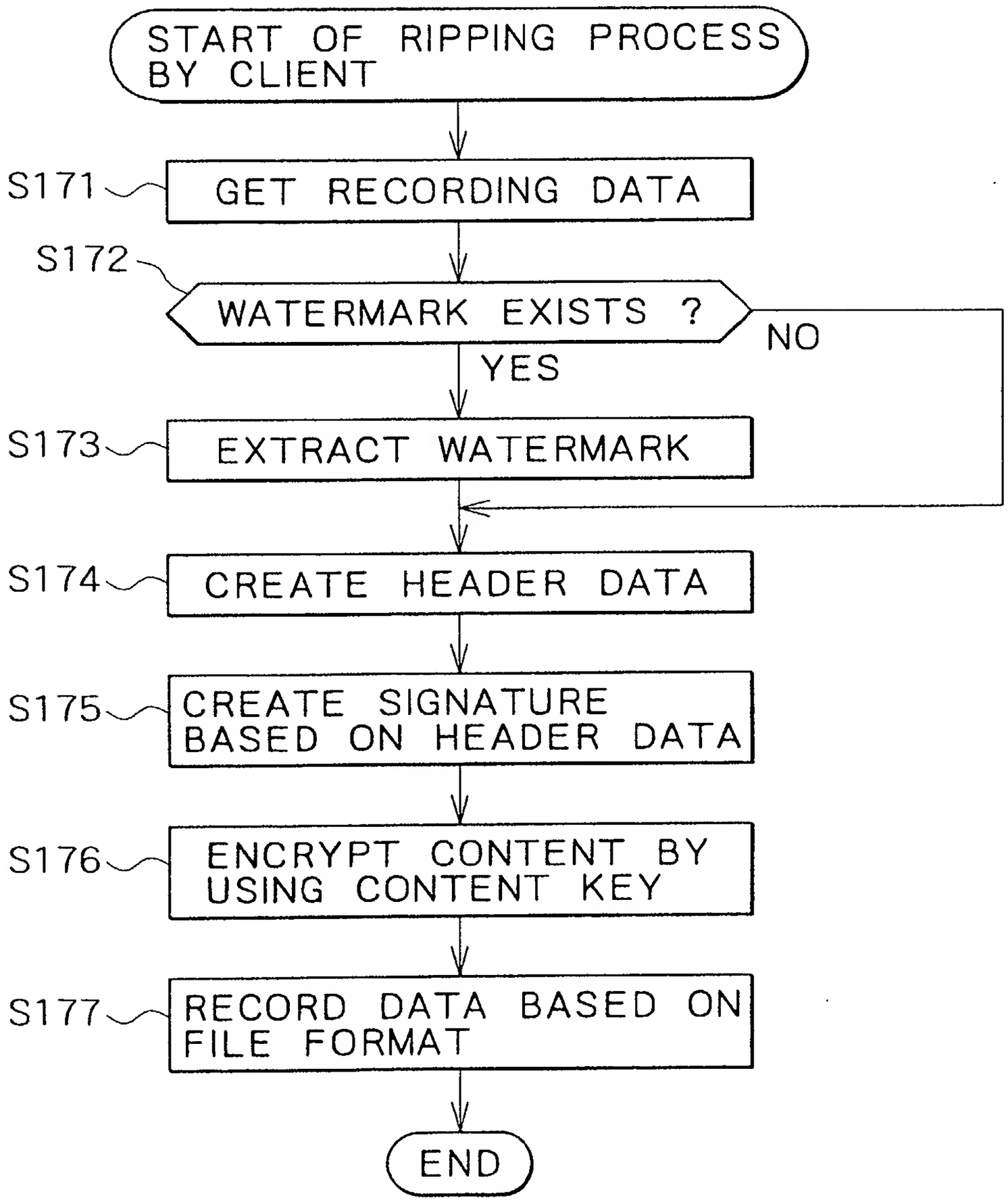

START OF RIPPING PROCESS BY CLIENT
S171
GET RECORDING DATA
S172
WATERMARK EXISTS ?
NO
YES
S173
EXTRACT WATERMARK
S174
CREATE HEADER DATA
S175
CREATE SIGNATURE BASED ON HEADER DATA
S176
ENCRYPT CONTENT BY USING CONTENT KEY
S177
RECORD DATA BASED ON FILE FORMAT
END

FIG. 24

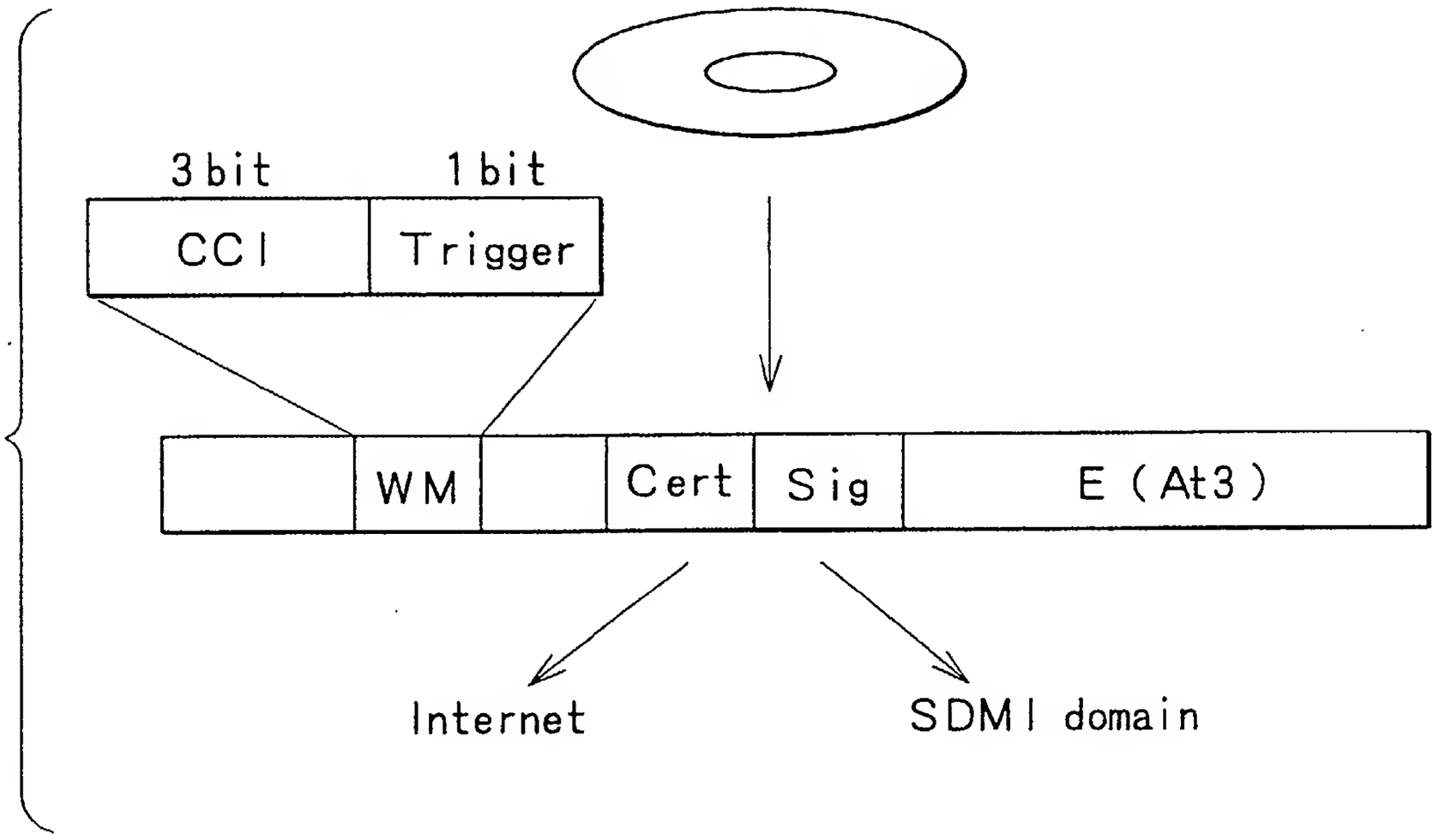

3 bit
1 bit
CCI
Trigger
WM
Cert
Sig
E (At3)
Internet
SDMI domain

FIG. 25

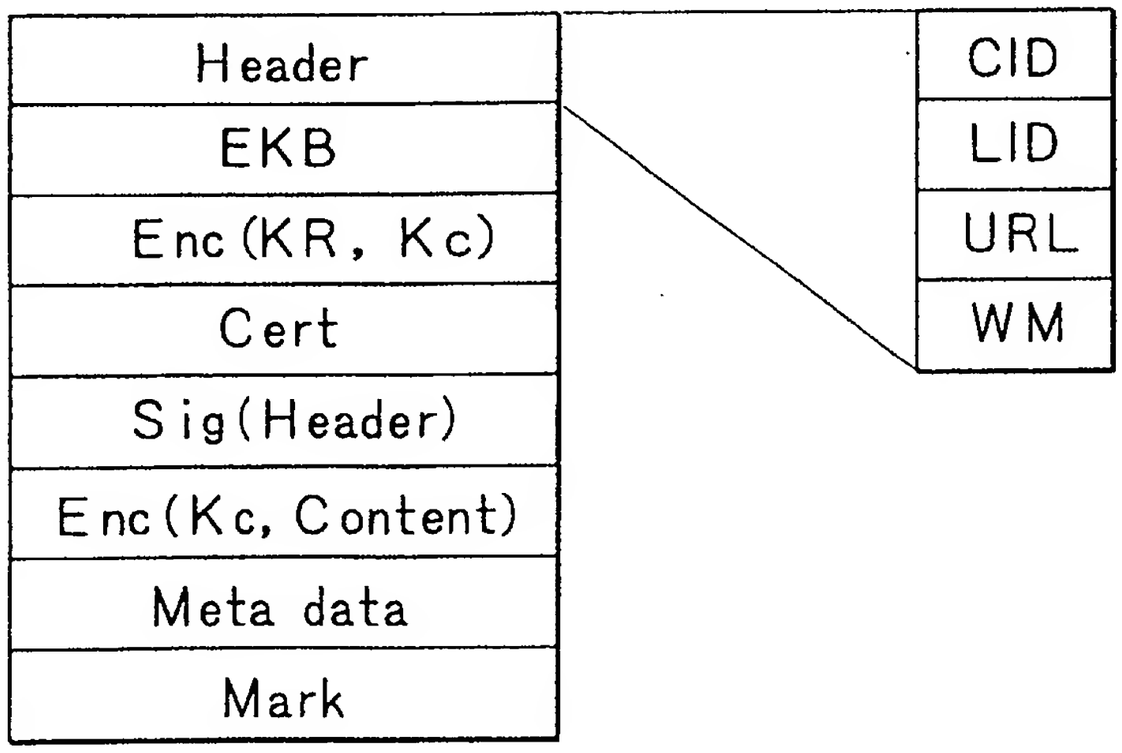

Header
EKB
Enc(KR, Kc)
Cert
Sig(Header)
Enc(Kc, Content)
Meta data
Mark
CID
LID
URL
WM

FIG. 26

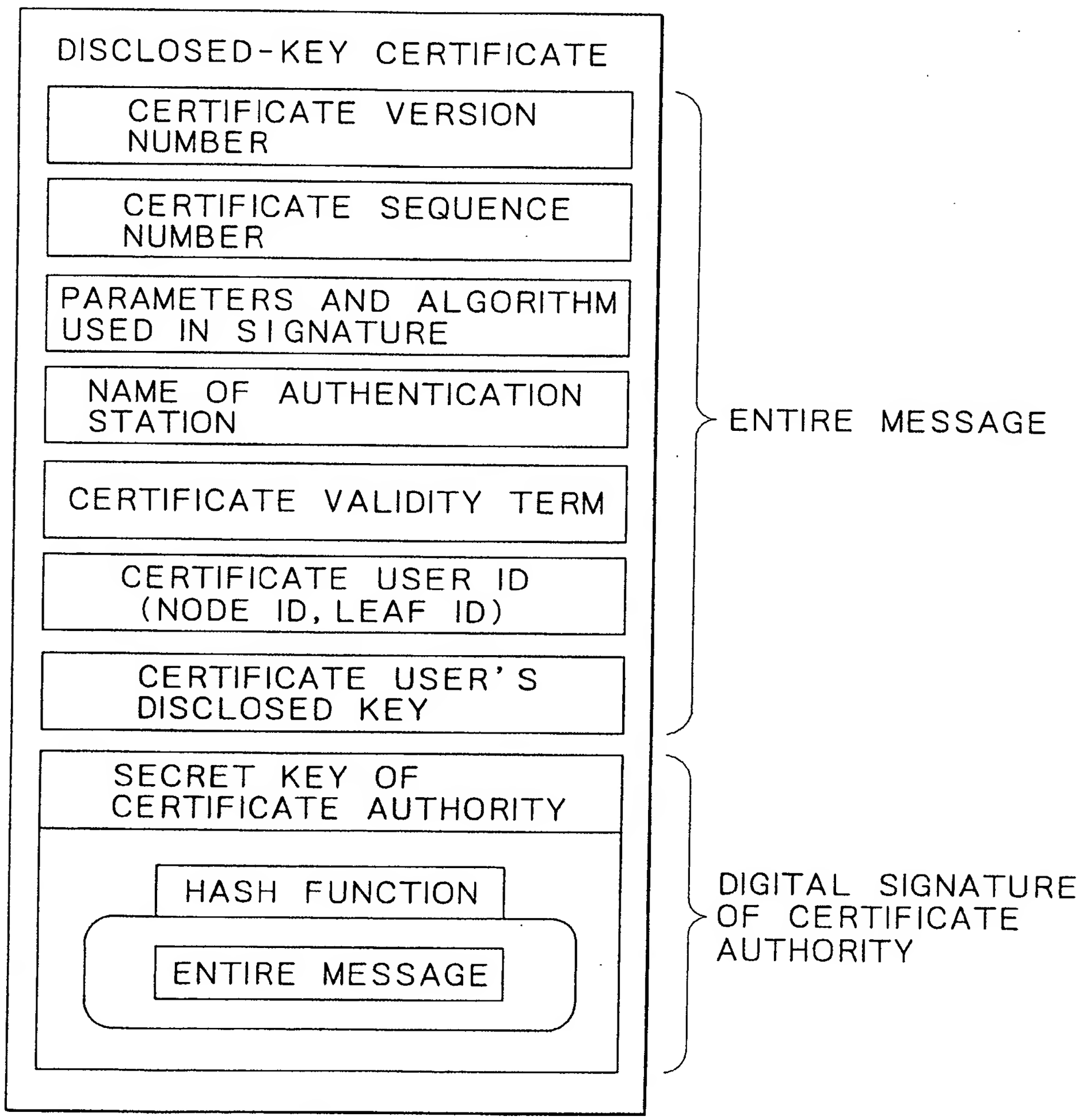

DISCLOSED-KEY CERTIFICATE
CERTIFICATE VERSION NUMBER
CERTIFICATE SEQUENCE NUMBER
PARAMETERS AND ALGORITHM USED IN SIGNATURE
NAME OF AUTHENTICATION STATION
CERTIFICATE VALIDITY TERM
CERTIFICATE USER ID (NODE ID, LEAF ID)
CERTIFICATE USER'S DISCLOSED KEY
SECRET KEY OF CERTIFICATE AUTHORITY
HASH FUNCTION
ENTIRE MESSAGE
ENTIRE MESSAGE
DIGITAL SIGNATURE OF CERTIFICATE AUTHORITY

FIG. 27

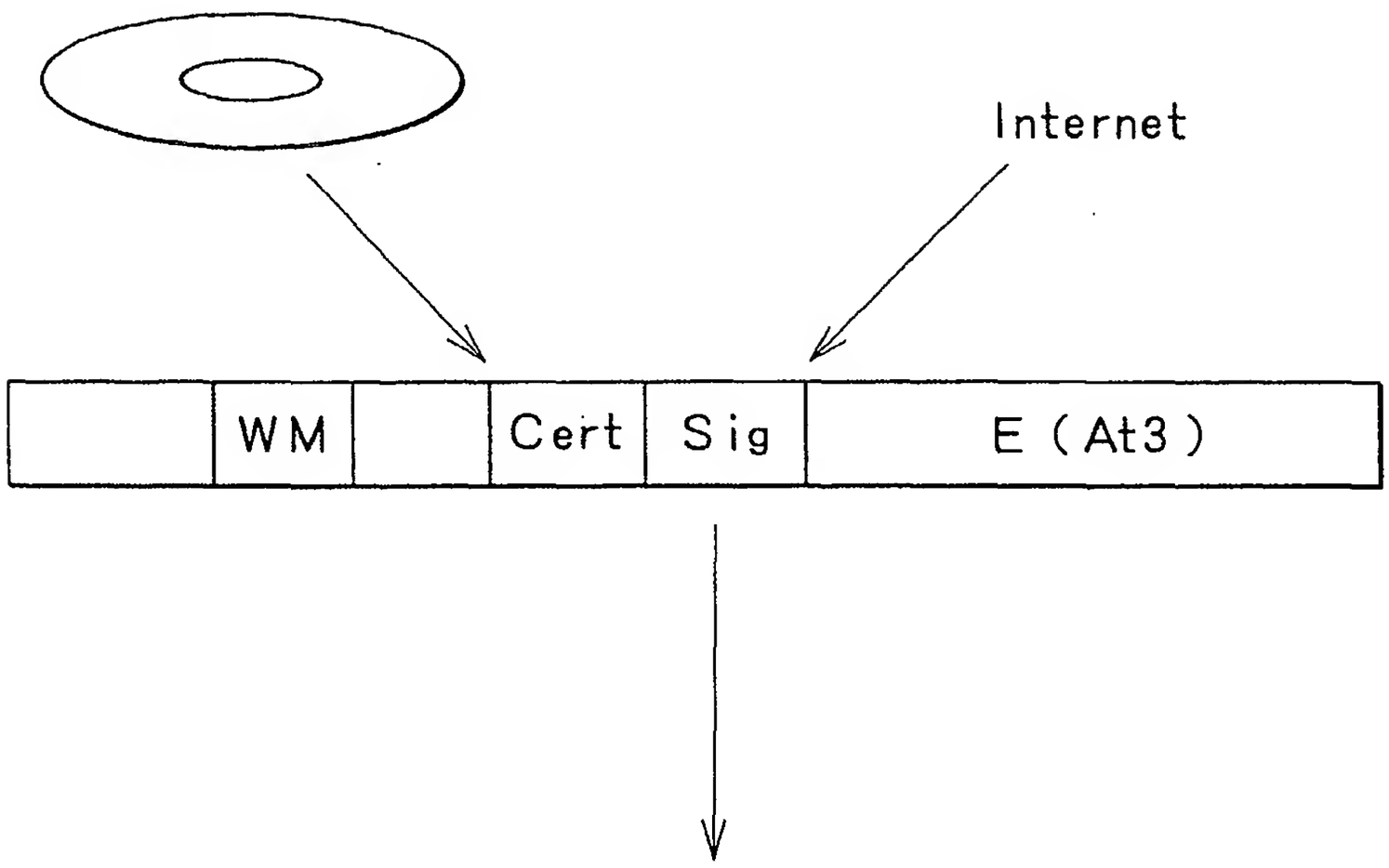


FIG. 28

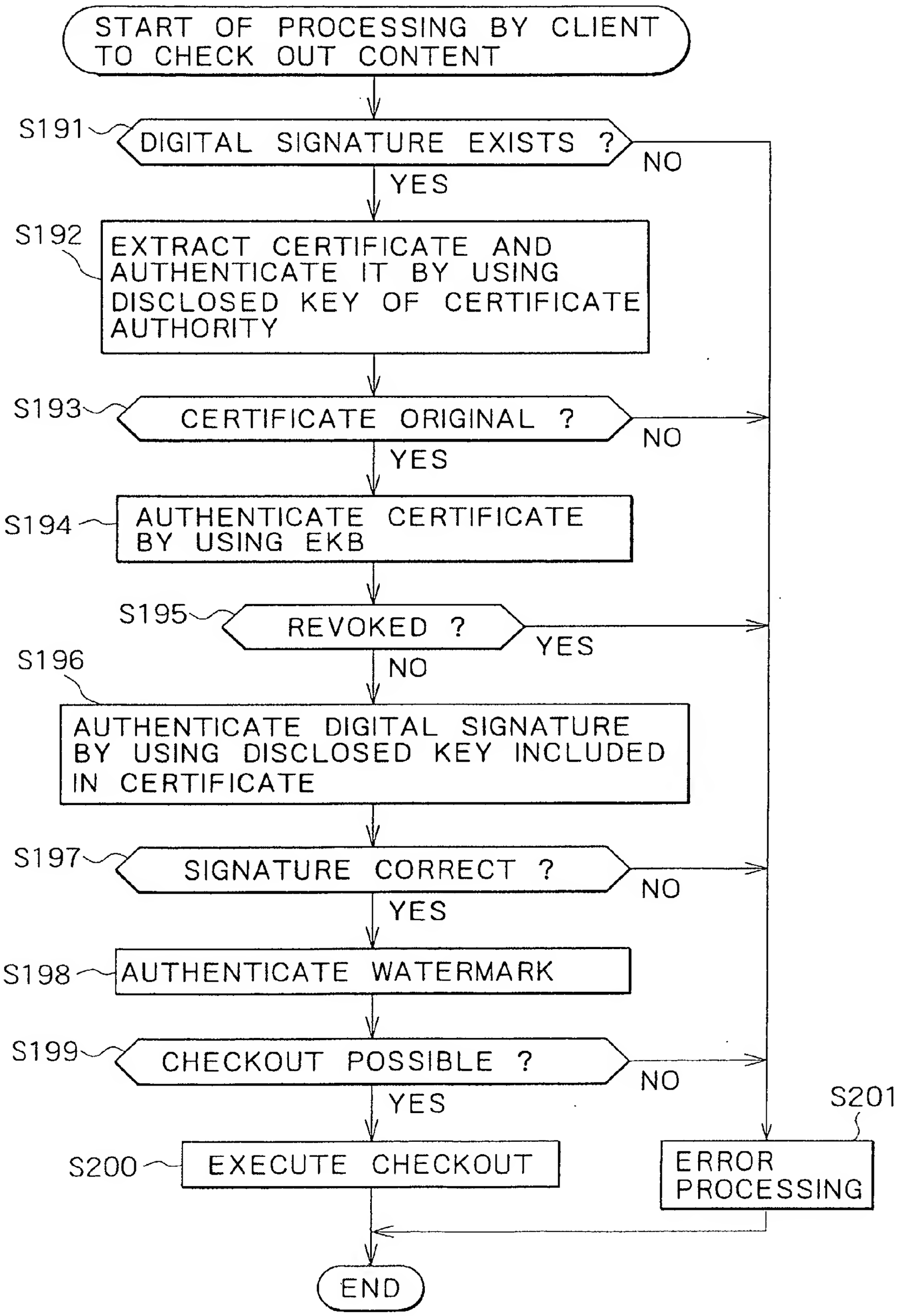

START OF PROCESSING BY CLIENT TO CHECK OUT CONTENT
S191
DIGITAL SIGNATURE EXISTS ?
NO
YES
S192
EXTRACT CERTIFICATE AND AUTHENTICATE IT BY USING DISCLOSED KEY OF CERTIFICATE AUTHORITY
S193
CERTIFICATE ORIGINAL ?
NO
YES
S194
AUTHENTICATE CERTIFICATE BY USING EKB
S195
REVOKED ?
YES
NO
S196
AUTHENTICATE DIGITAL SIGNATURE BY USING DISCLOSED KEY INCLUDED IN CERTIFICATE
S197
SIGNATURE CORRECT ?
NO
YES
S198
AUTHENTICATE WATERMARK
S199
CHECKOUT POSSIBLE ?
NO
YES
S200
EXECUTE CHECKOUT
S201
ERROR PROCESSING
END

FIG. 29

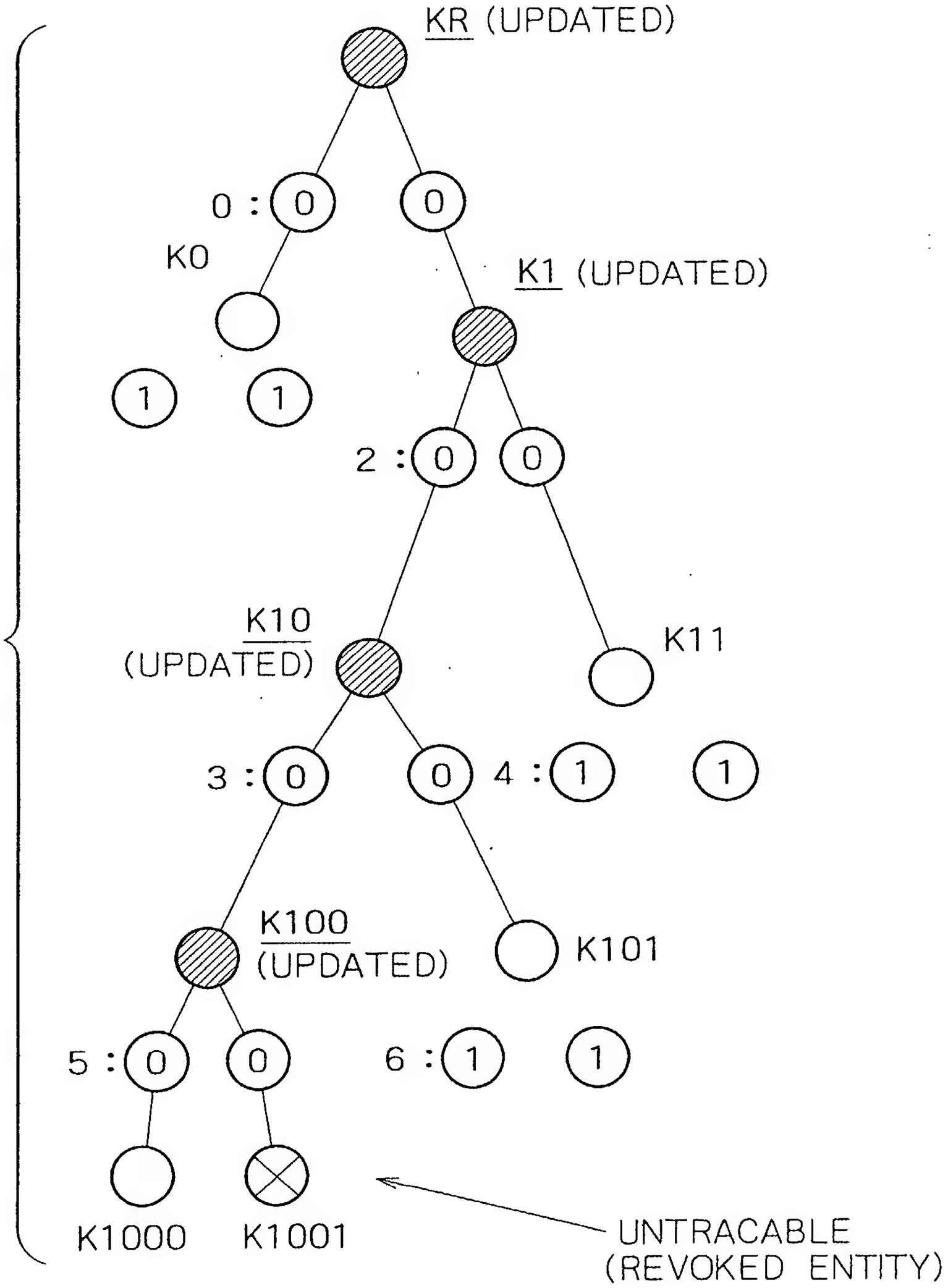

KR (UPDATED)
0 : 0
0
K0
K1 (UPDATED)
1
1
2 : 0
0
K10
(UPDATED)
K11
3 : 0
0
4 : 1
1
K100
(UPDATED)
K101
5 : 0
0
6 : 1
1
K1000
K1001
UNTRACABLE
(REVOKED ENTITY)

FIG.30

DATA PORTION AND TAGS OF EKB
(ENABLING KEY BLOCK)

DATA (ENCRYPTED KEYS)	Enc(K0, K(t)R, Enc(K(t)1, K(t)R) Enc(K(t)10, K(t)1), Enc(K11, K(t)1) Enc(K(t)100, K(t)10), Enc(K101, K(t)10) Enc(K1000, K(t)100)
TAGS	0 : {0,0}, 1 : {1,1}, 2 : {0,0}, 3 : {0,0} 4 : {1,1}, 5 : {0,1}, 6 : {1,1}

↕

{L TAG, R TAG}

L TAG AND R TAG ARE EACH 0 TO INDICATE EXISTENCE OF DATA OR 1 OTHERWISE

FIG.31

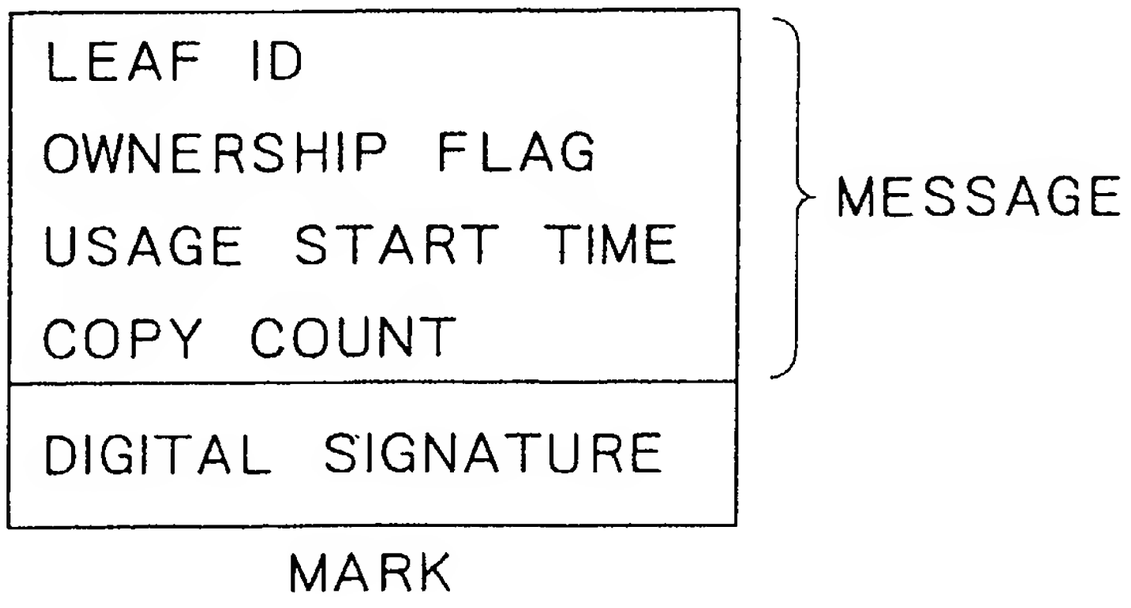


FIG. 32

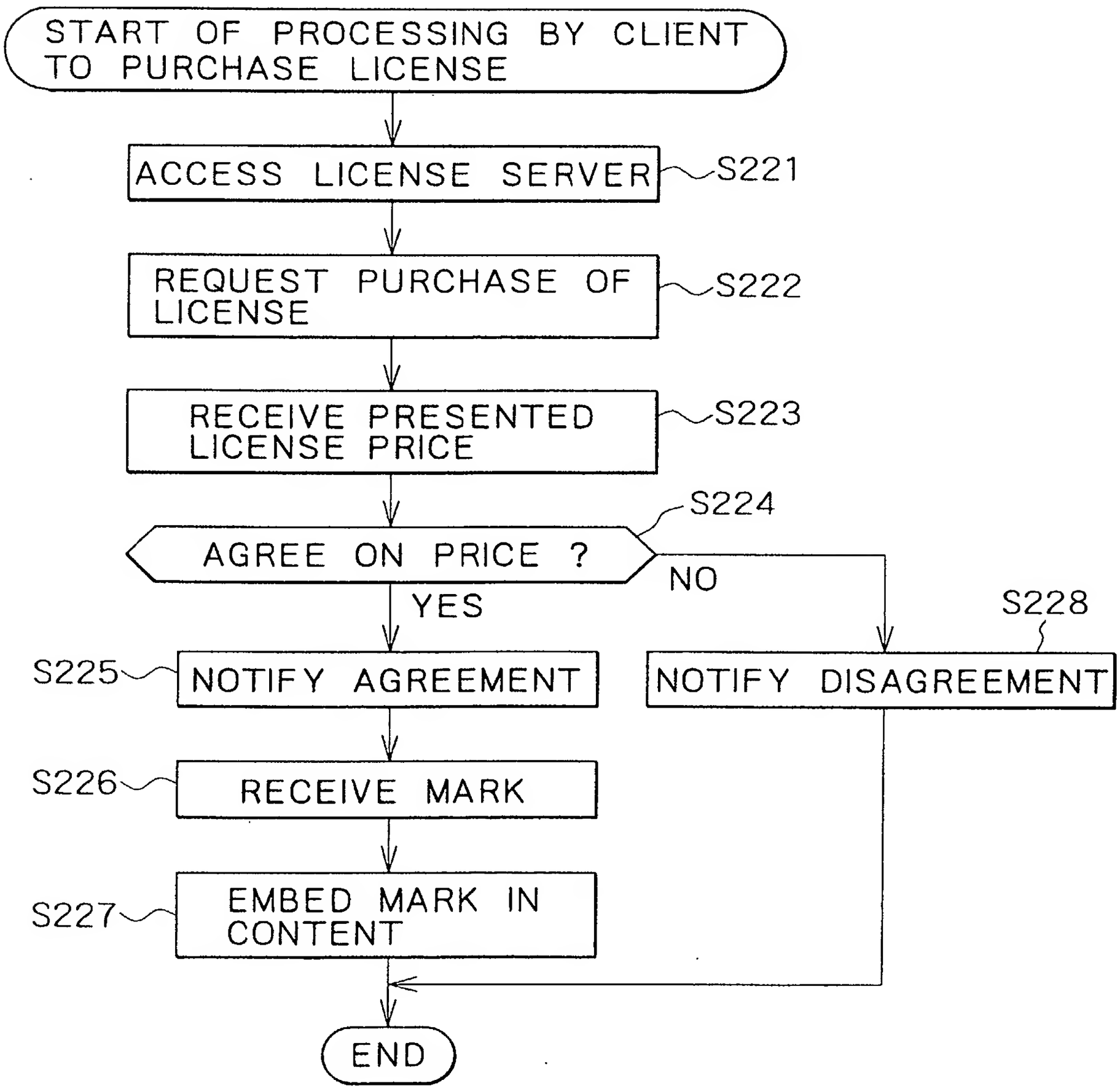

START OF PROCESSING BY CLIENT TO PURCHASE LICENSE
ACCESS LICENSE SERVER
S221
REQUEST PURCHASE OF LICENSE
S222
RECEIVE PRESENTED LICENSE PRICE
S223
S224
AGREE ON PRICE ?
NO
YES
S228
S225
NOTIFY AGREEMENT
NOTIFY DISAGREEMENT
S226
RECEIVE MARK
S227
EMBED MARK IN CONTENT
END

FIG. 33

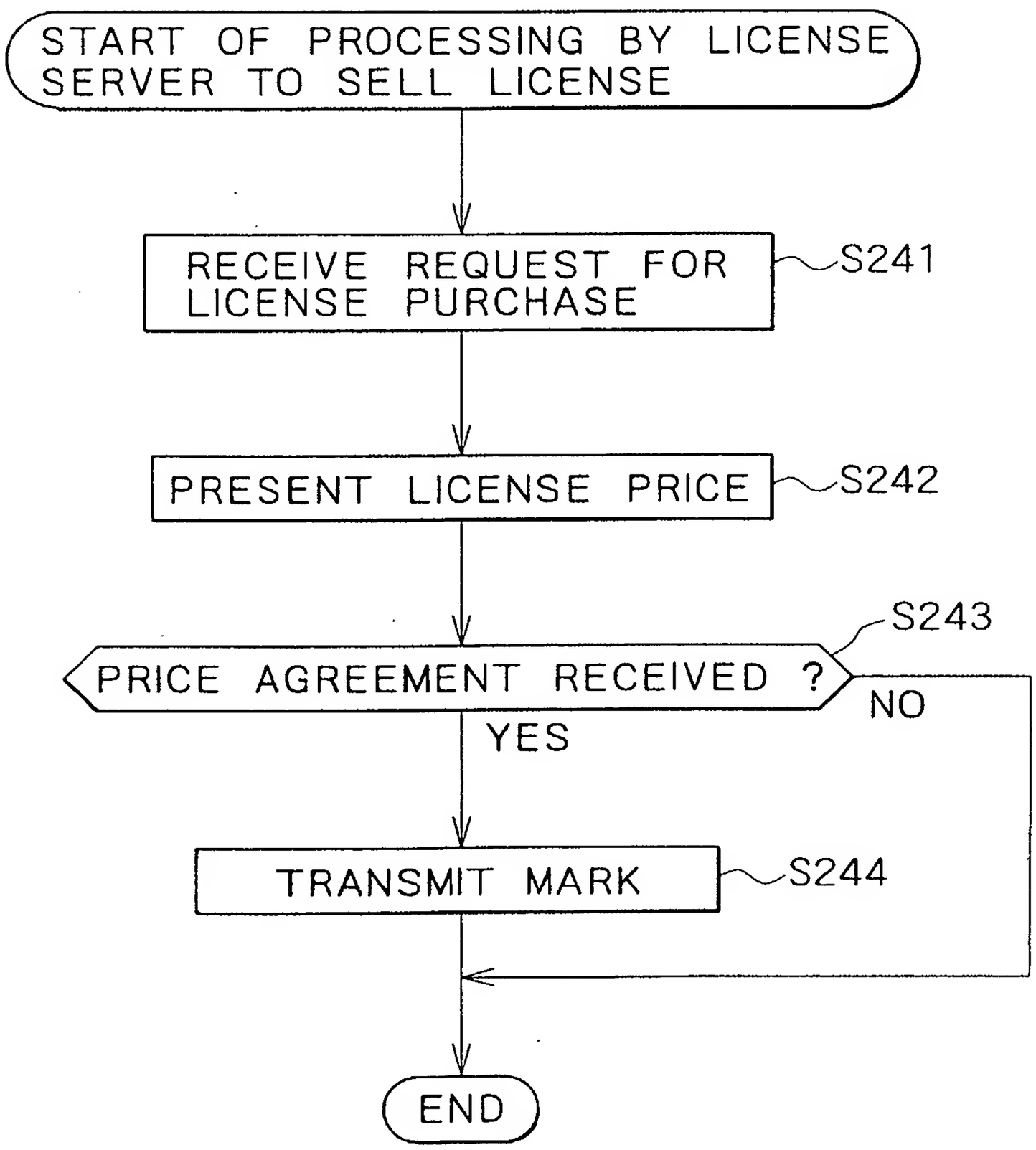

START OF PROCESSING BY LICENSE SERVER TO SELL LICENSE
RECEIVE REQUEST FOR LICENSE PURCHASE
S241
PRESENT LICENSE PRICE
S242
PRICE AGREEMENT RECEIVED ?
S243
NO
YES
TRANSMIT MARK
S244
END

FIG.34

$$Mark = \{LeafID, Own, Sig_S(LeafID, Own)\}$$

FIG.35

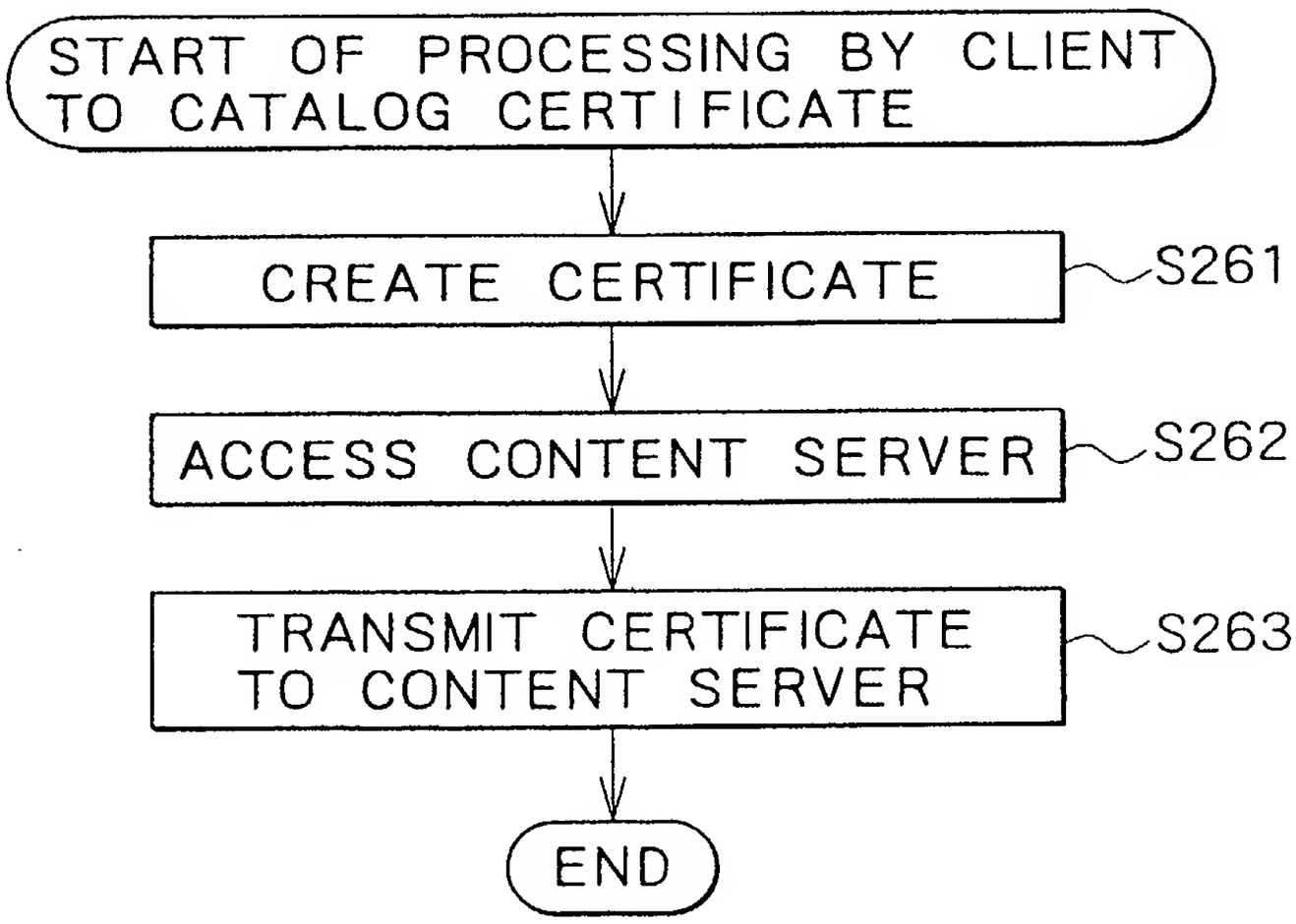


FIG. 36

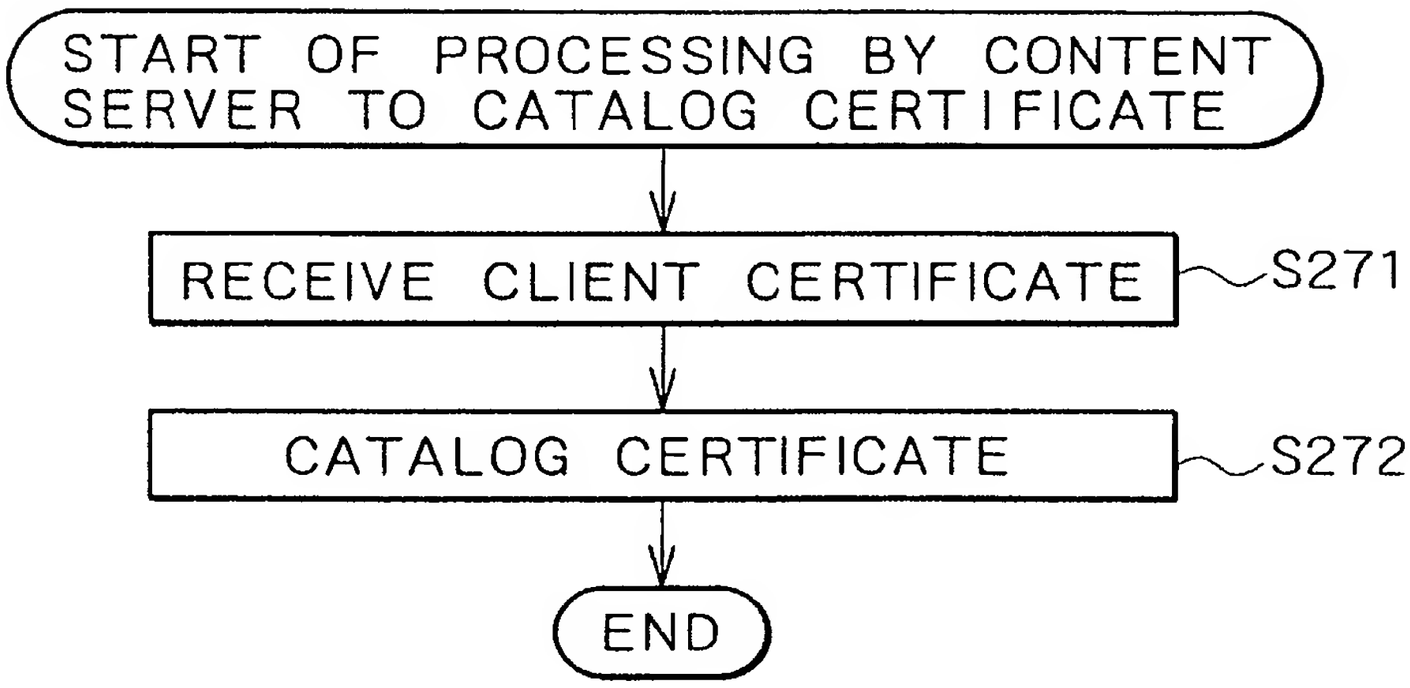

START OF PROCESSING BY CONTENT SERVER TO CATALOG CERTIFICATE
RECEIVE CLIENT CERTIFICATE
S271
CATALOG CERTIFICATE
S272
END

FIG. 37

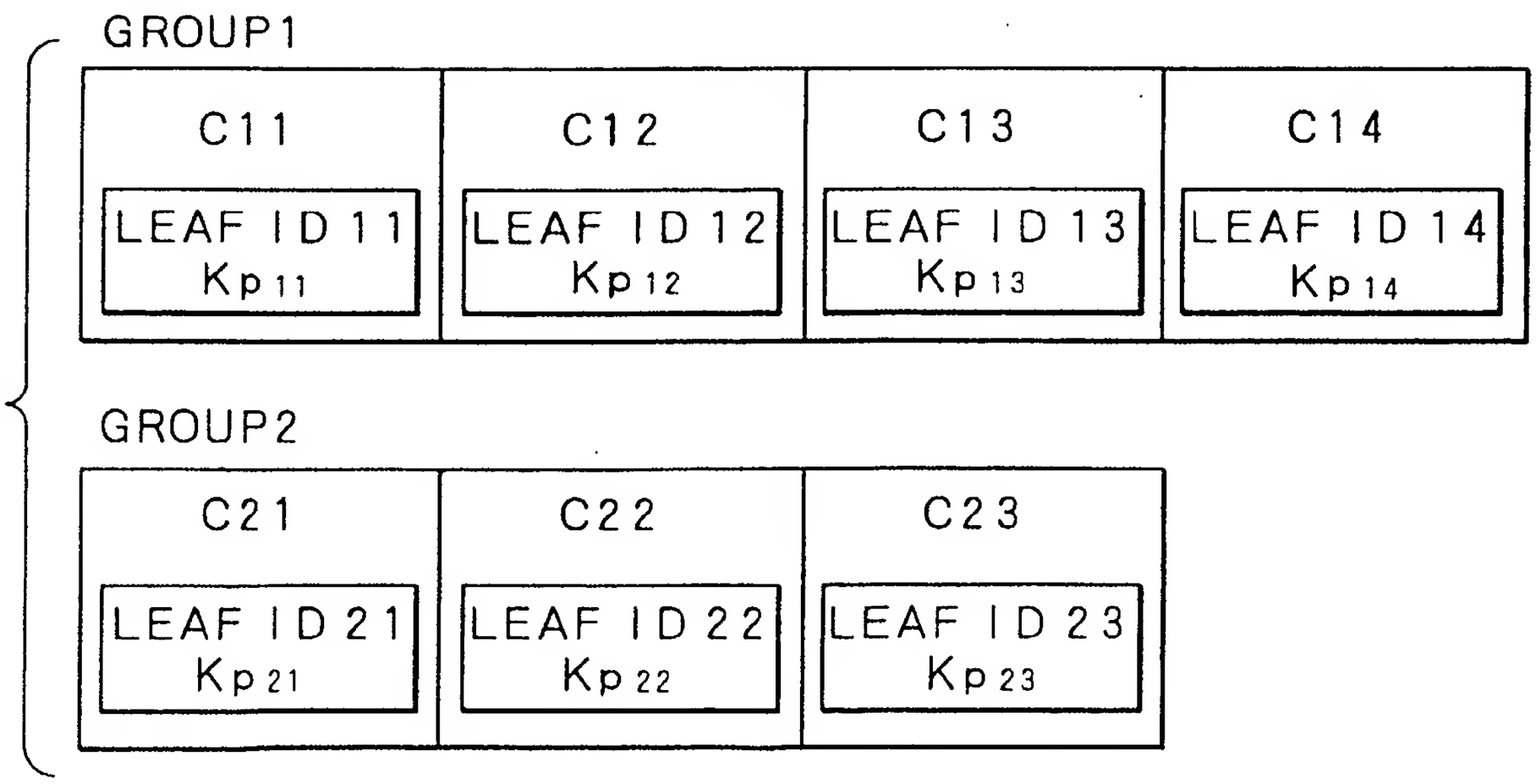

GROUP1
C11
LEAF ID 11
Kp11
C12
LEAF ID 12
Kp12
C13
LEAF ID 13
Kp13
C14
LEAF ID 14
Kp14
GROUP2
C21
LEAF ID 21
Kp21
C22
LEAF ID 22
Kp22
C23
LEAF ID 23
Kp23

FIG. 38

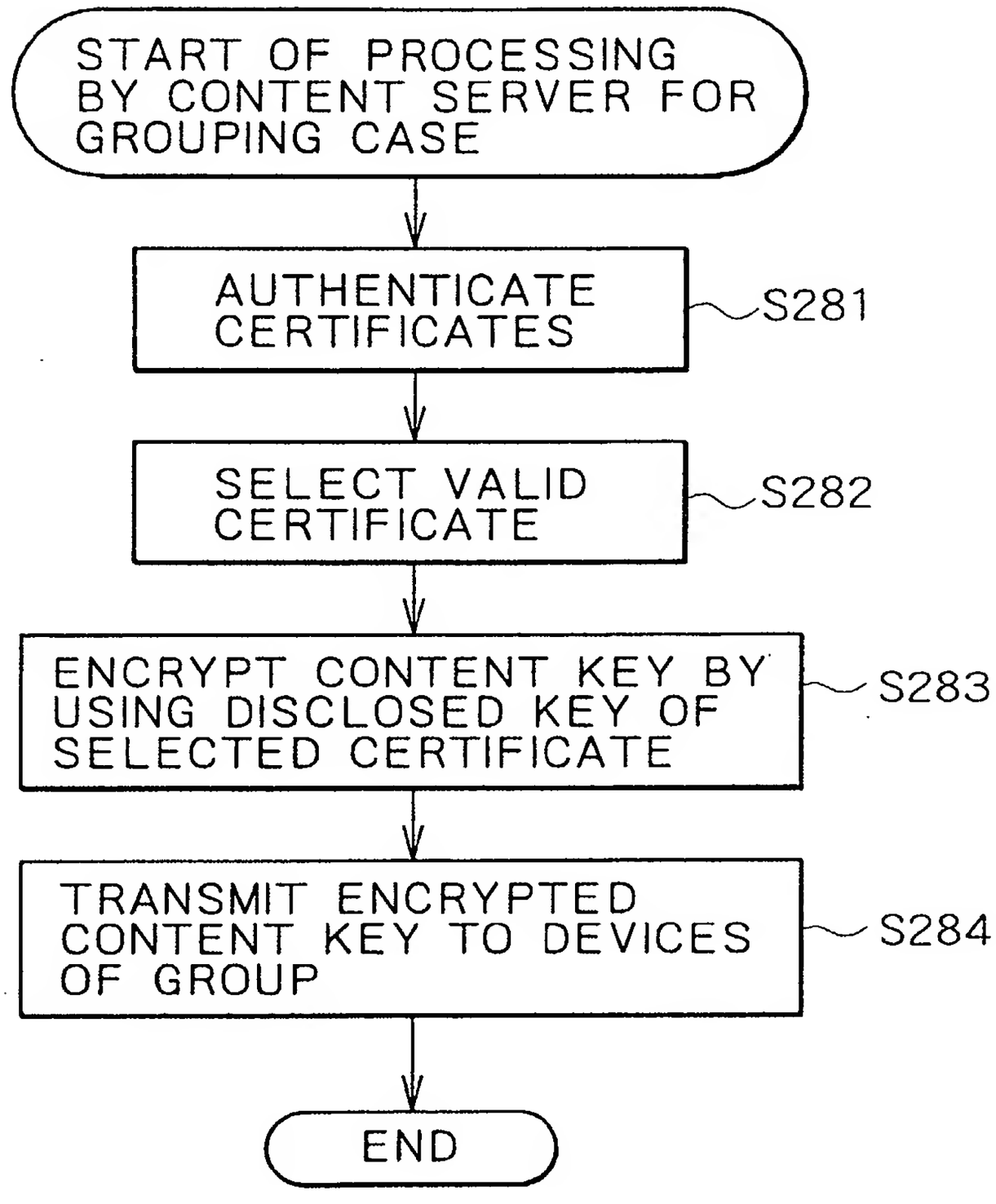


FIG. 39

$Enc(Kp_{11}, Kc), Enc(Kp_{12}, Kc), Enc(Kp_{13}, Kc)$

FIG.40

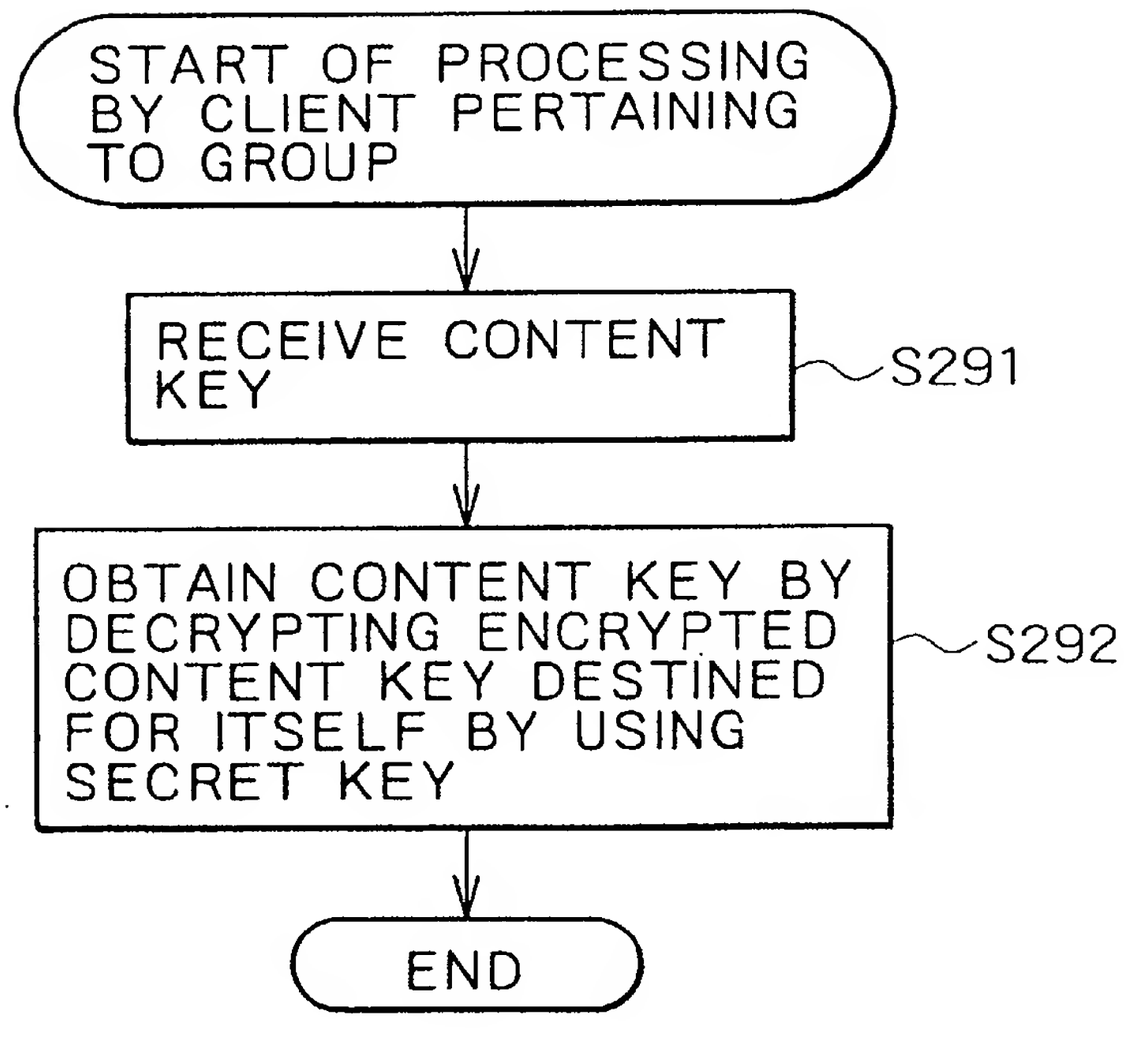

START OF PROCESSING BY CLIENT PERTAINING TO GROUP
RECEIVE CONTENT KEY
S291
OBTAIN CONTENT KEY BY DECRYPTING ENCRYPTED CONTENT KEY DESTINED FOR ITSELF BY USING SECRET KEY
S292
END

FIG. 41

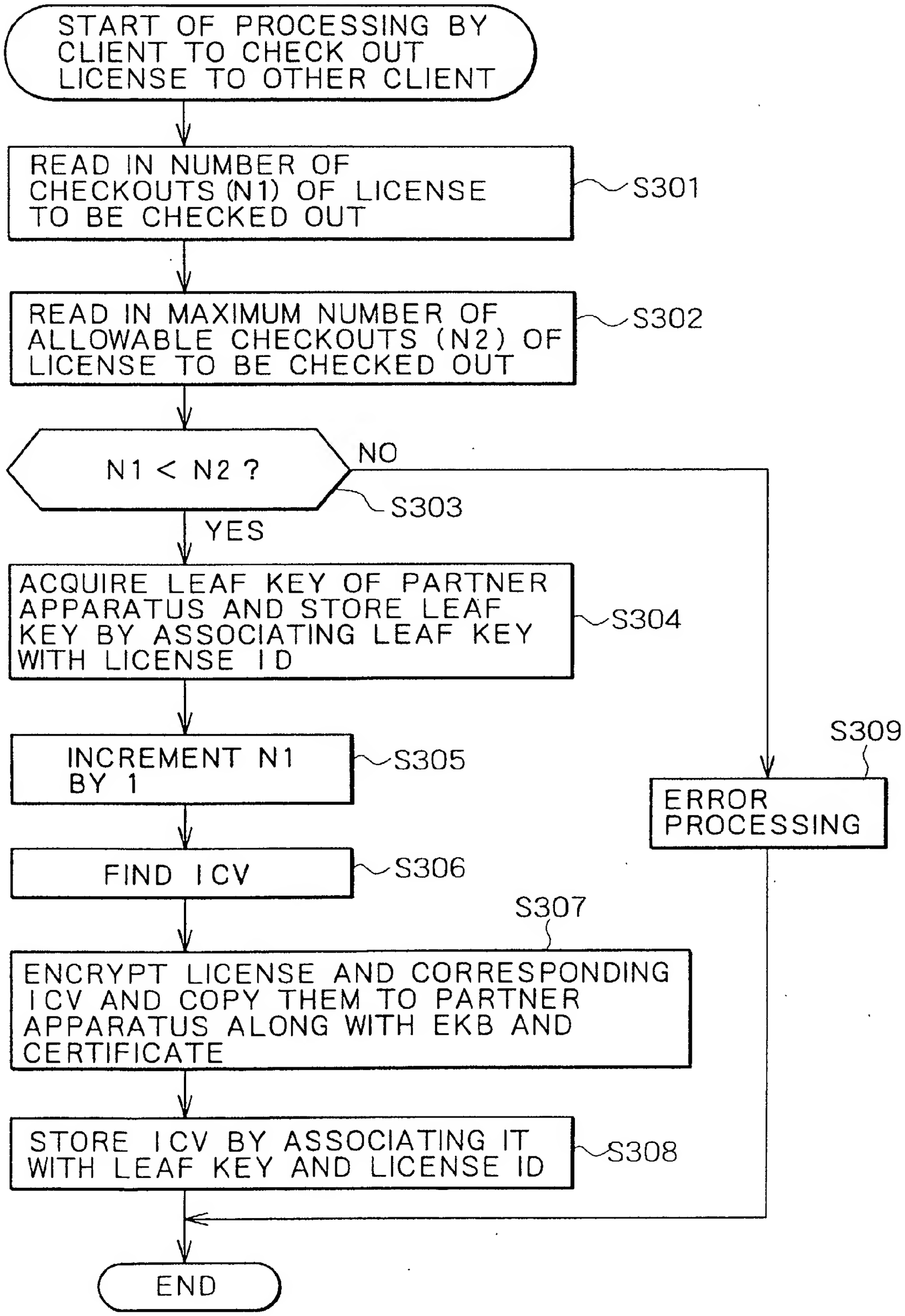

START OF PROCESSING BY CLIENT TO CHECK OUT LICENSE TO OTHER CLIENT
READ IN NUMBER OF CHECKOUTS (N1) OF LICENSE TO BE CHECKED OUT
S301
READ IN MAXIMUM NUMBER OF ALLOWABLE CHECKOUTS (N2) OF LICENSE TO BE CHECKED OUT
S302
N1 < N2 ?
NO
S303
YES
ACQUIRE LEAF KEY OF PARTNER APPARATUS AND STORE LEAF KEY BY ASSOCIATING LEAF KEY WITH LICENSE ID
S304
INCREMENT N1 BY 1
S305
FIND ICV
S306
S307
ENCRYPT LICENSE AND CORRESPONDING ICV AND COPY THEM TO PARTNER APPARATUS ALONG WITH EKB AND CERTIFICATE
STORE ICV BY ASSOCIATING IT WITH LEAF KEY AND LICENSE ID
S308
S309
ERROR PROCESSING
END

FIG.42

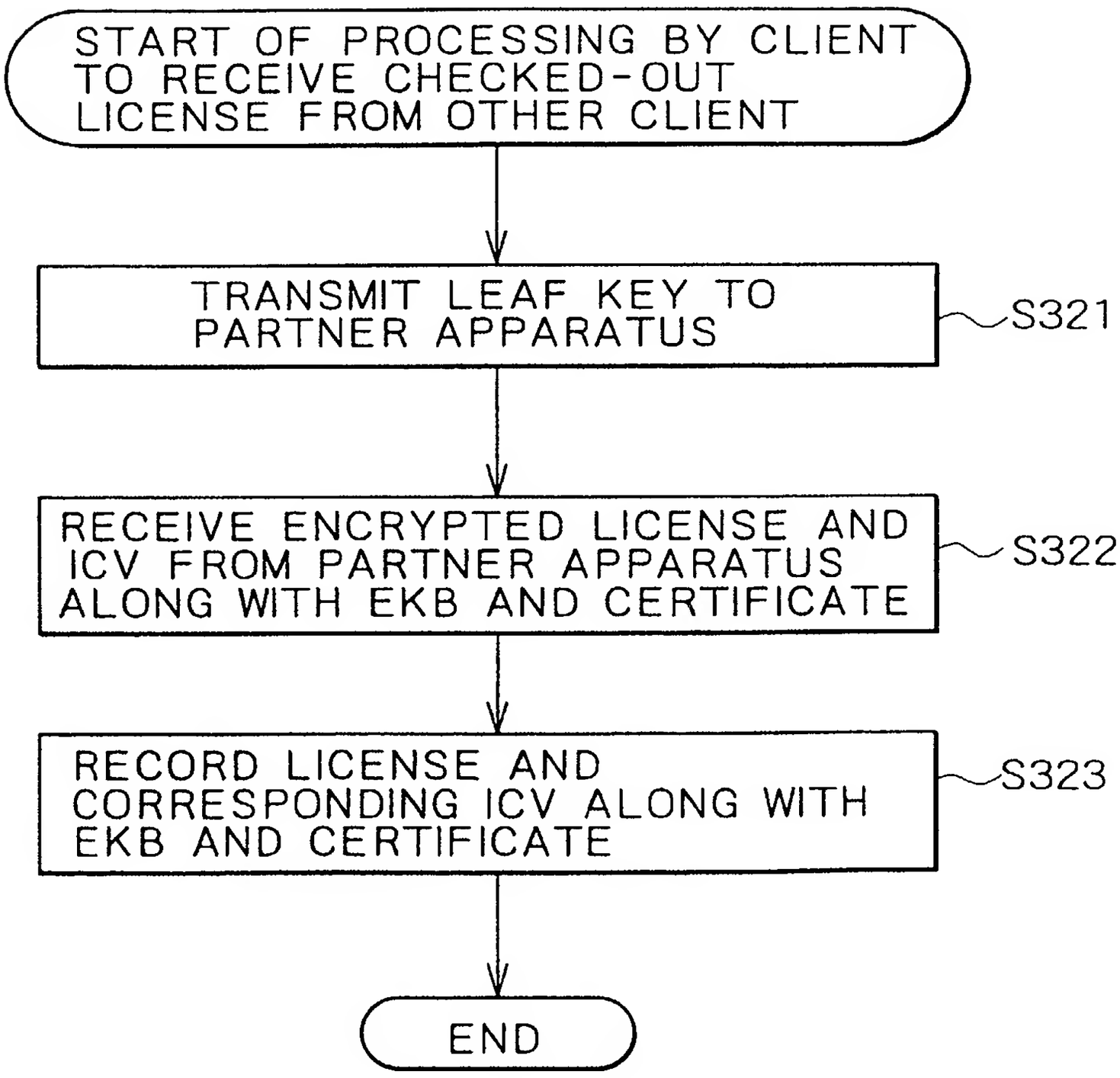

START OF PROCESSING BY CLIENT TO RECEIVE CHECKED-OUT LICENSE FROM OTHER CLIENT
TRANSMIT LEAF KEY TO PARTNER APPARATUS
S321
RECEIVE ENCRYPTED LICENSE AND ICV FROM PARTNER APPARATUS ALONG WITH EKB AND CERTIFICATE
S322
RECORD LICENSE AND CORRESPONDING ICV ALONG WITH EKB AND CERTIFICATE
S323
END

FIG. 43

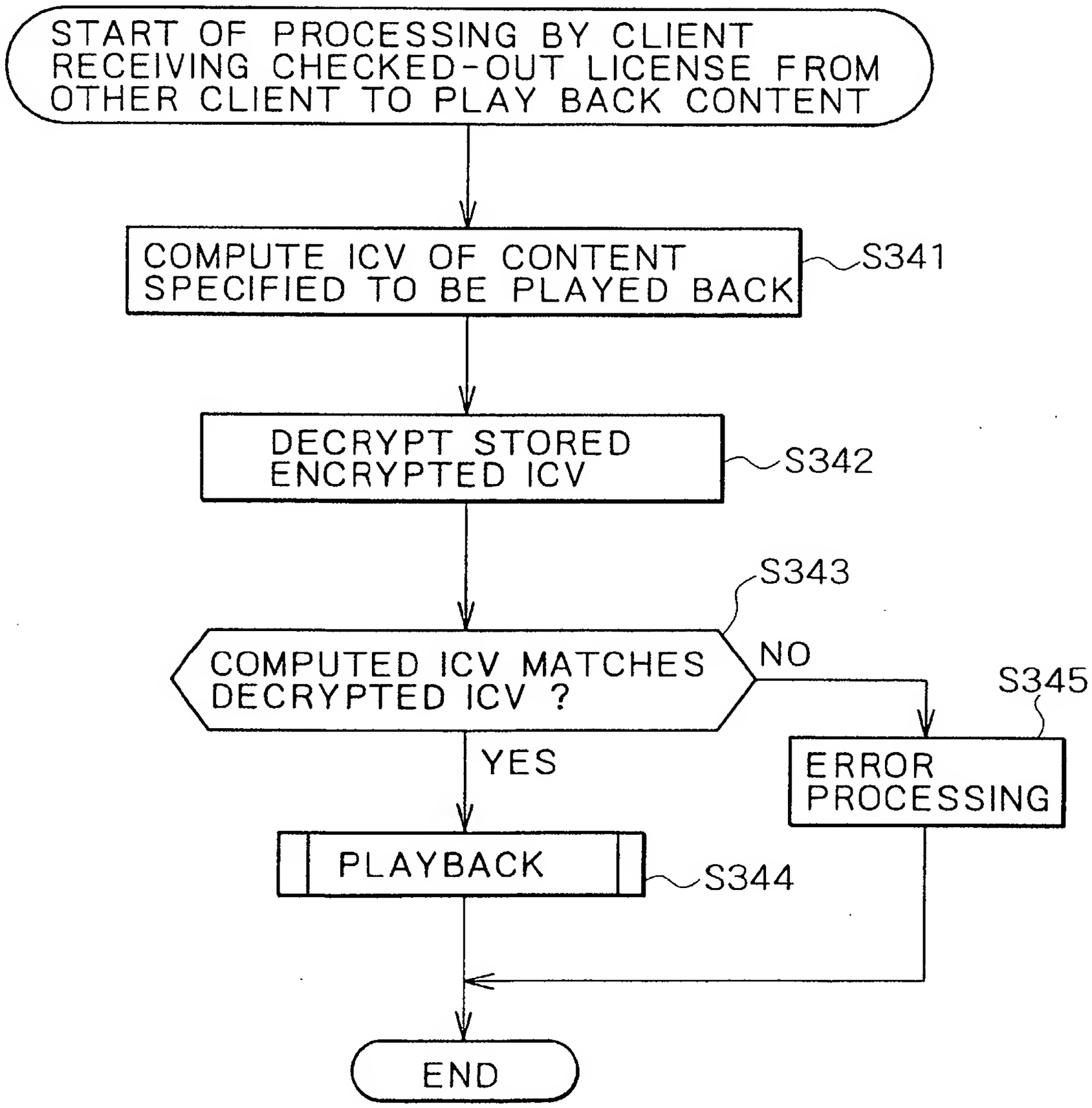

START OF PROCESSING BY CLIENT RECEIVING CHECKED-OUT LICENSE FROM OTHER CLIENT TO PLAY BACK CONTENT
COMPUTE ICV OF CONTENT SPECIFIED TO BE PLAYED BACK
S341
DECRYPT STORED ENCRYPTED ICV
S342
S343
COMPUTED ICV MATCHES DECRYPTED ICV ?
NO
S345
ERROR PROCESSING
YES
PLAYBACK
S344
END

FIG. 44

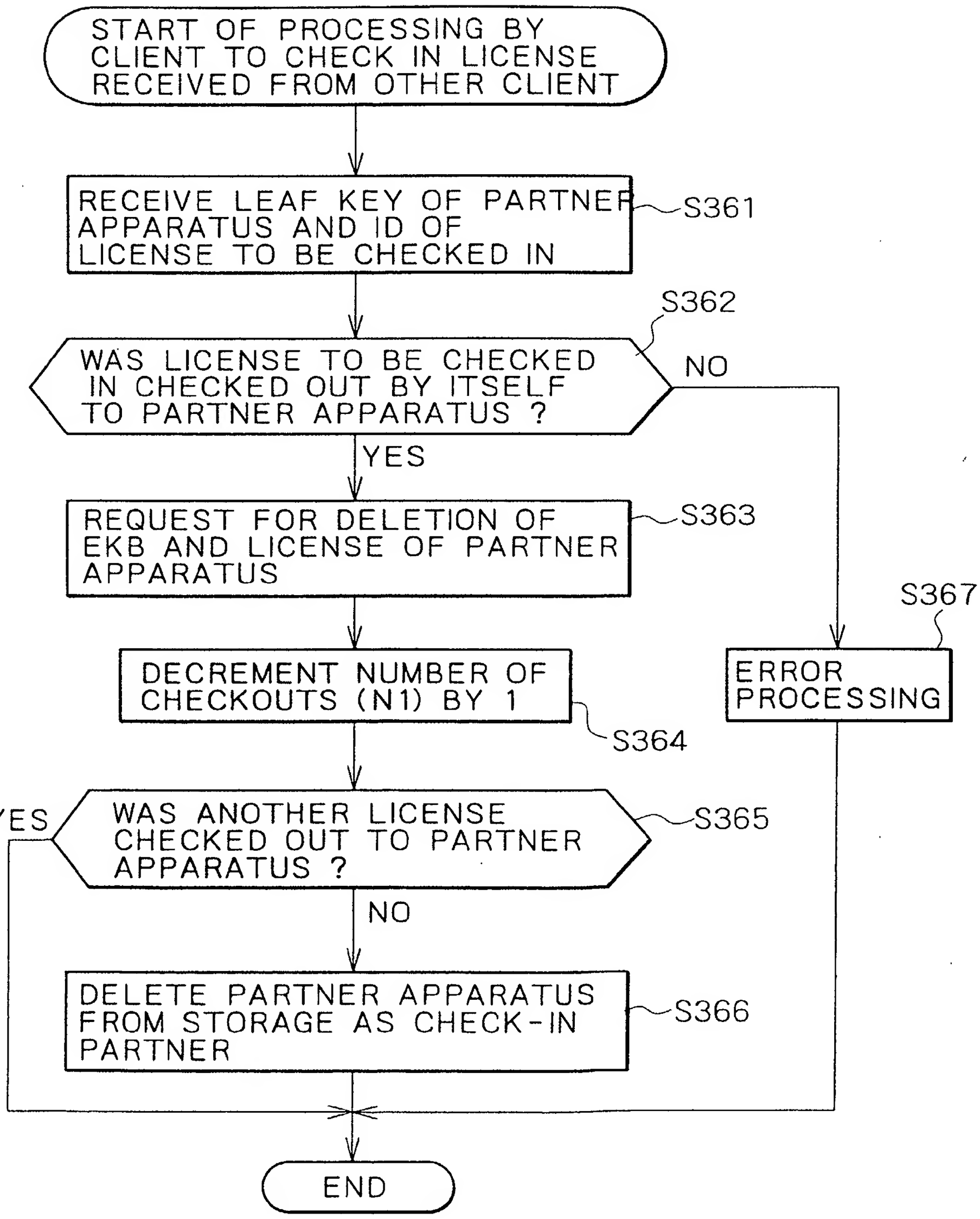

START OF PROCESSING BY CLIENT TO CHECK IN LICENSE RECEIVED FROM OTHER CLIENT
RECEIVE LEAF KEY OF PARTNER APPARATUS AND ID OF LICENSE TO BE CHECKED IN
S361
WAS LICENSE TO BE CHECKED IN CHECKED OUT BY ITSELF TO PARTNER APPARATUS ?
S362
NO
YES
REQUEST FOR DELETION OF EKB AND LICENSE OF PARTNER APPARATUS
S363
DECREMENT NUMBER OF CHECKOUTS (N1) BY 1
S364
ERROR PROCESSING
S367
WAS ANOTHER LICENSE CHECKED OUT TO PARTNER APPARATUS ?
S365
YES
NO
DELETE PARTNER APPARATUS FROM STORAGE AS CHECK-IN PARTNER
S366
END

FIG. 45

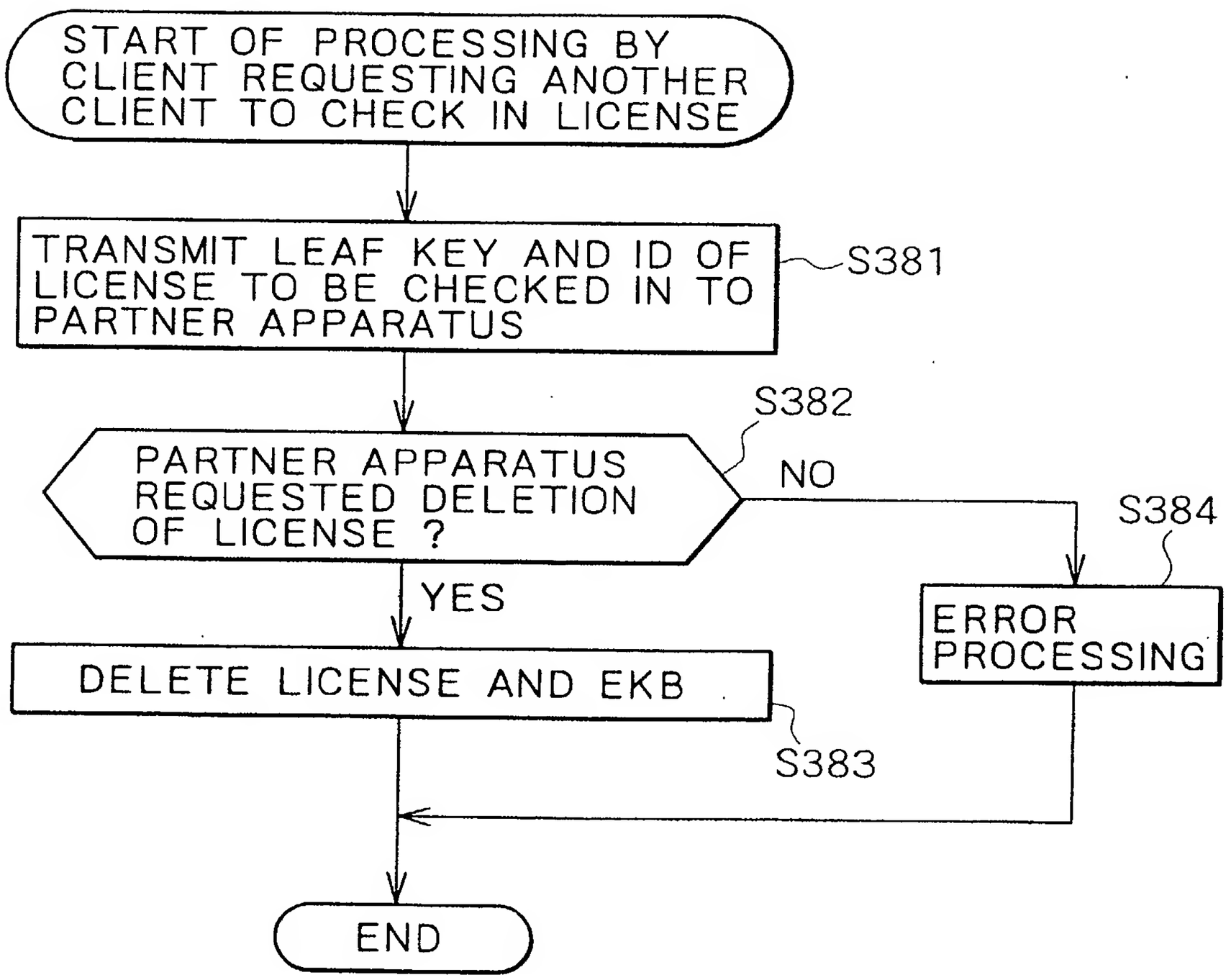

START OF PROCESSING BY CLIENT REQUESTING ANOTHER CLIENT TO CHECK IN LICENSE
TRANSMIT LEAF KEY AND ID OF LICENSE TO BE CHECKED IN TO PARTNER APPARATUS
S381
S382
PARTNER APPARATUS REQUESTED DELETION OF LICENSE ?
NO
S384
ERROR PROCESSING
YES
DELETE LICENSE AND EKB
S383
END

FIG. 46

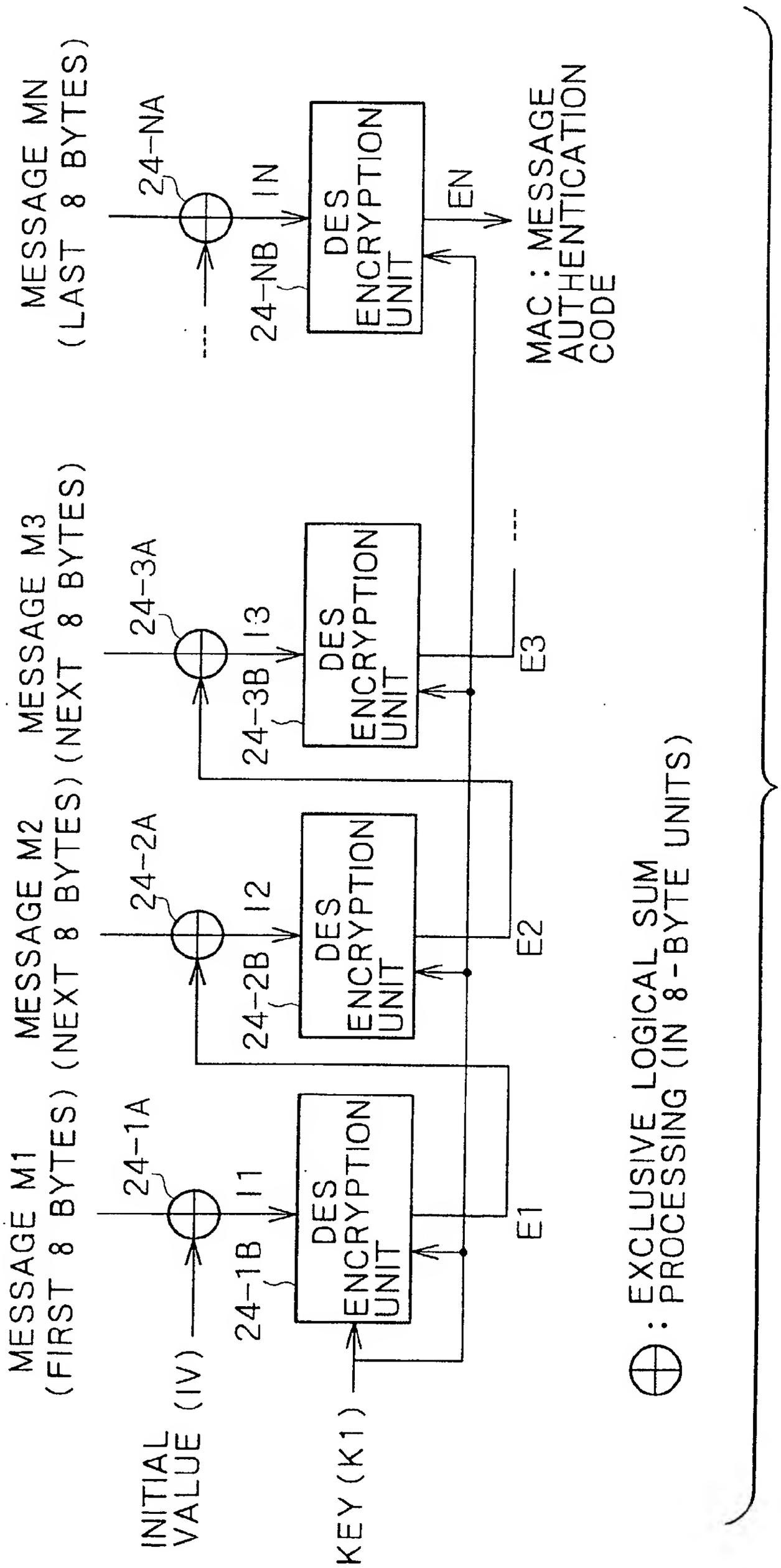

MESSAGE M1
(FIRST 8 BYTES)
MESSAGE M2
(NEXT 8 BYTES)
MESSAGE M3
(NEXT 8 BYTES)
MESSAGE MN
(LAST 8 BYTES)
INITIAL VALUE (IV)
KEY (K1)
24-1A
24-1B
24-2A
24-2B
24-3A
24-3B
24-NA
24-NB
I1
I2
I3
IN
DES ENCRYPTION UNIT
E1
E2
E3
EN
MAC : MESSAGE AUTHENTICATION CODE
⊕ : EXCLUSIVE LOGICAL SUM PROCESSING (IN 8-BYTE UNITS)

FIG. 47

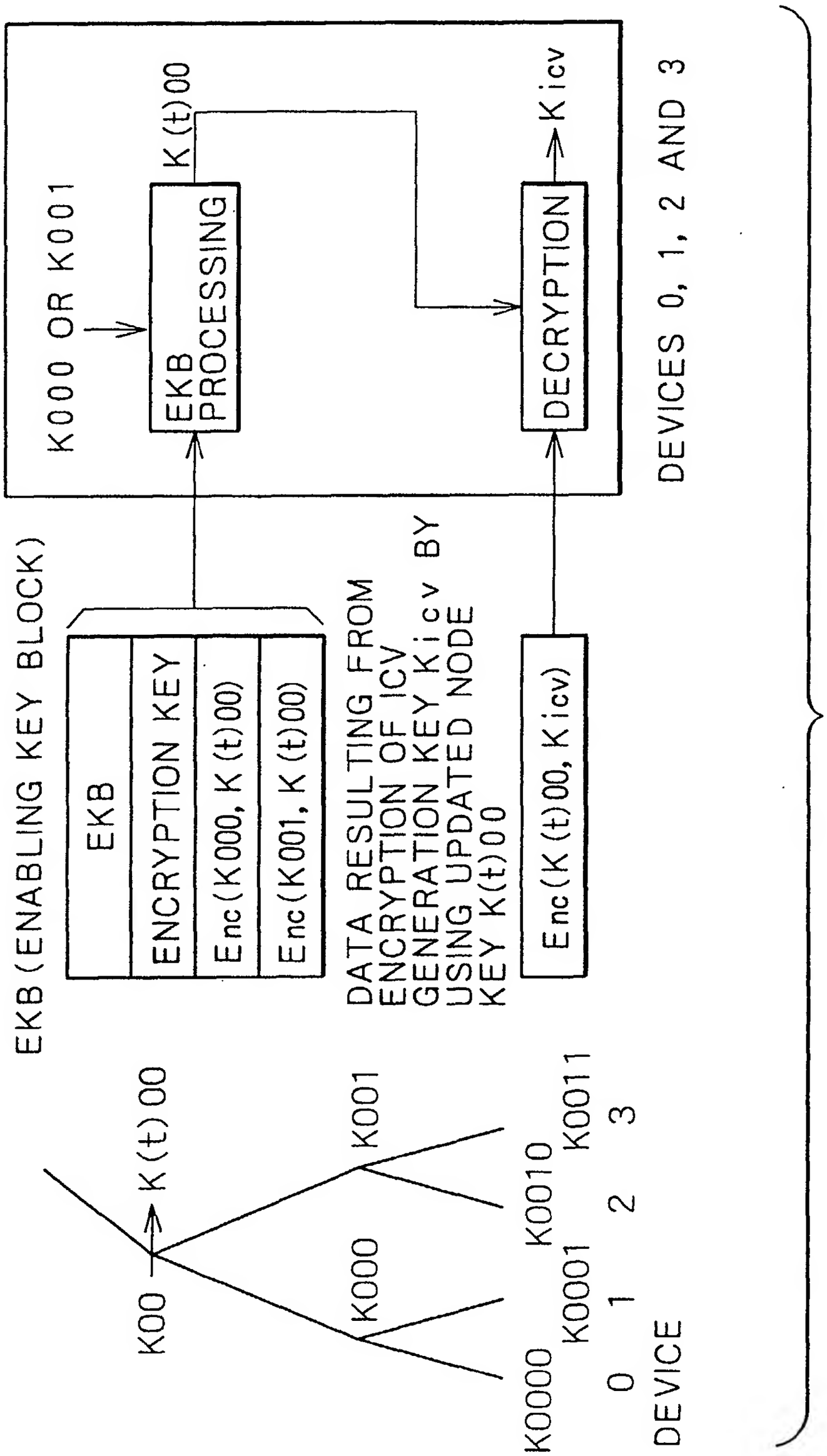
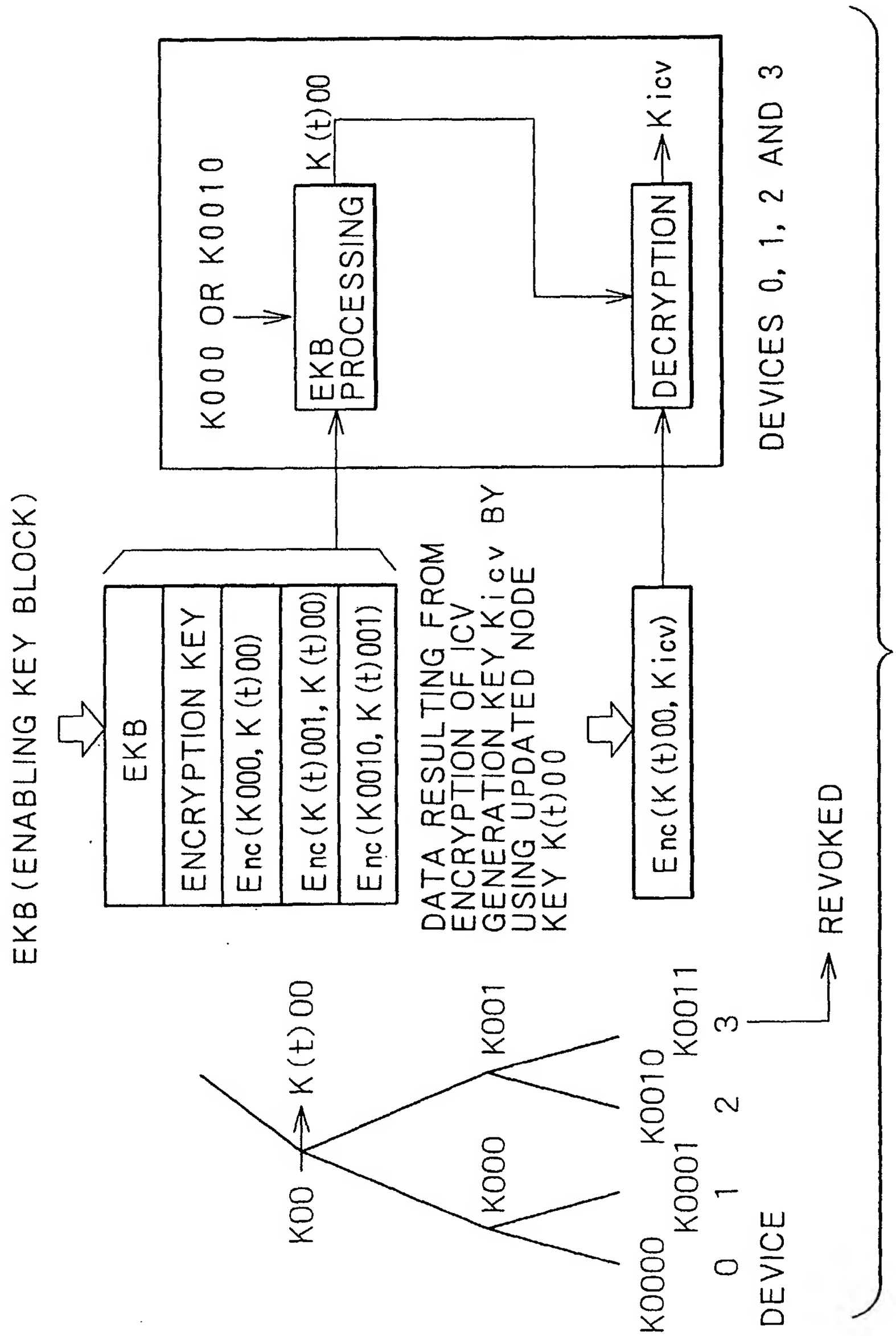

K000 OR K001
EKB PROCESSING
K(t)00
DECRYPTION
Kicv
DEVICES 0, 1, 2 AND 3
EKB (ENABLING KEY BLOCK)
EKB
ENCRYPTION KEY
Enc(K000, K(t)00)
Enc(K001, K(t)00)
DATA RESULTING FROM ENCRYPTION OF ICV GENERATION KEY Kicv BY USING UPDATED NODE KEY K(t)00
Enc(K(t)00, Kicv)
K00
K(t)00
K000
K001
K0000
K0001
K0010
K0011
0
1
2
3
DEVICE

FIG. 48


K000 OR K0010
EKB PROCESSING
K(t)00
DECRYPTION
K icv
DEVICES 0, 1, 2 AND 3
EKB (ENABLING KEY BLOCK)
EKB
ENCRYPTION KEY
Enc(K000, K(t)00)
Enc(K(t)001, K(t)00)
Enc(K0010, K(t)001)
DATA RESULTING FROM ENCRYPTION OF ICV GENERATION KEY K icv BY USING UPDATED NODE KEY K(t)00
Enc(K(t)00, K icv)
REVOKED
K00
K(t)00
K000
K001
K0000
K0001
K0010
K0011
0
1
2
3
DEVICE

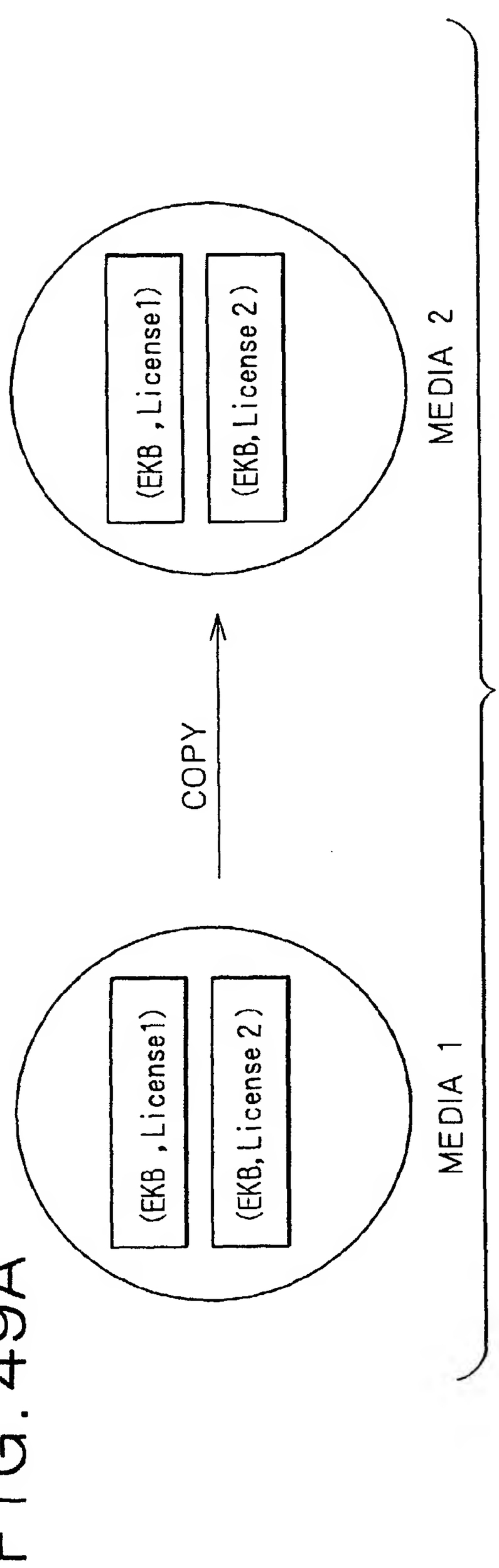


FIG. 49A

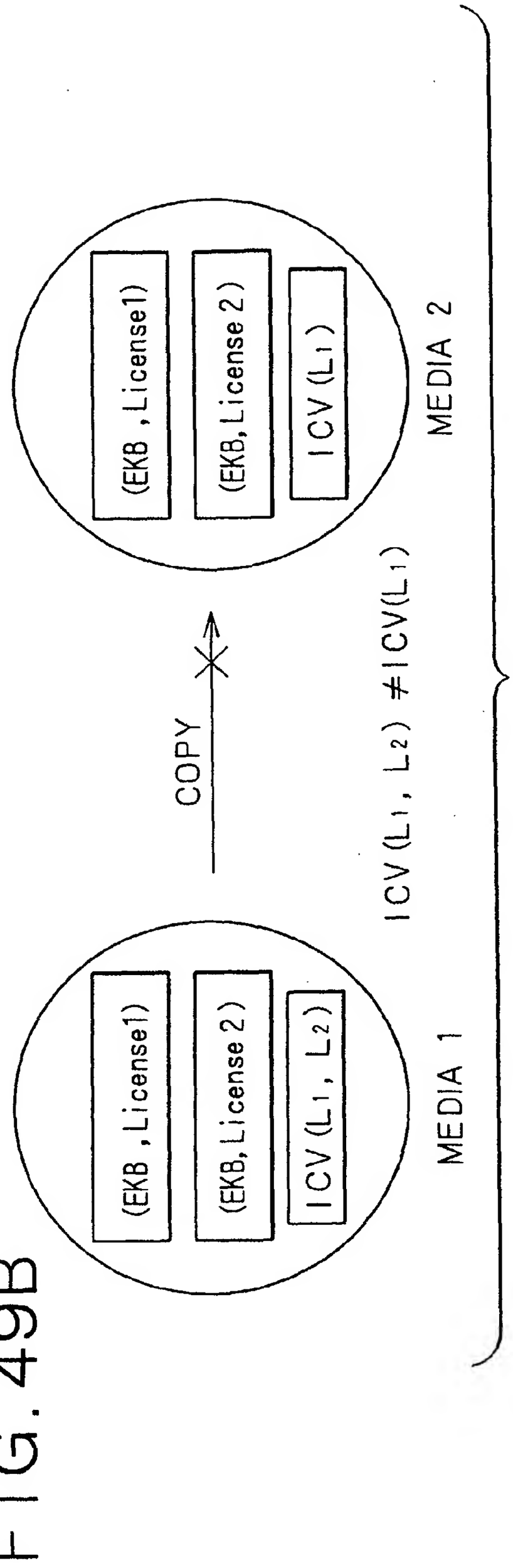
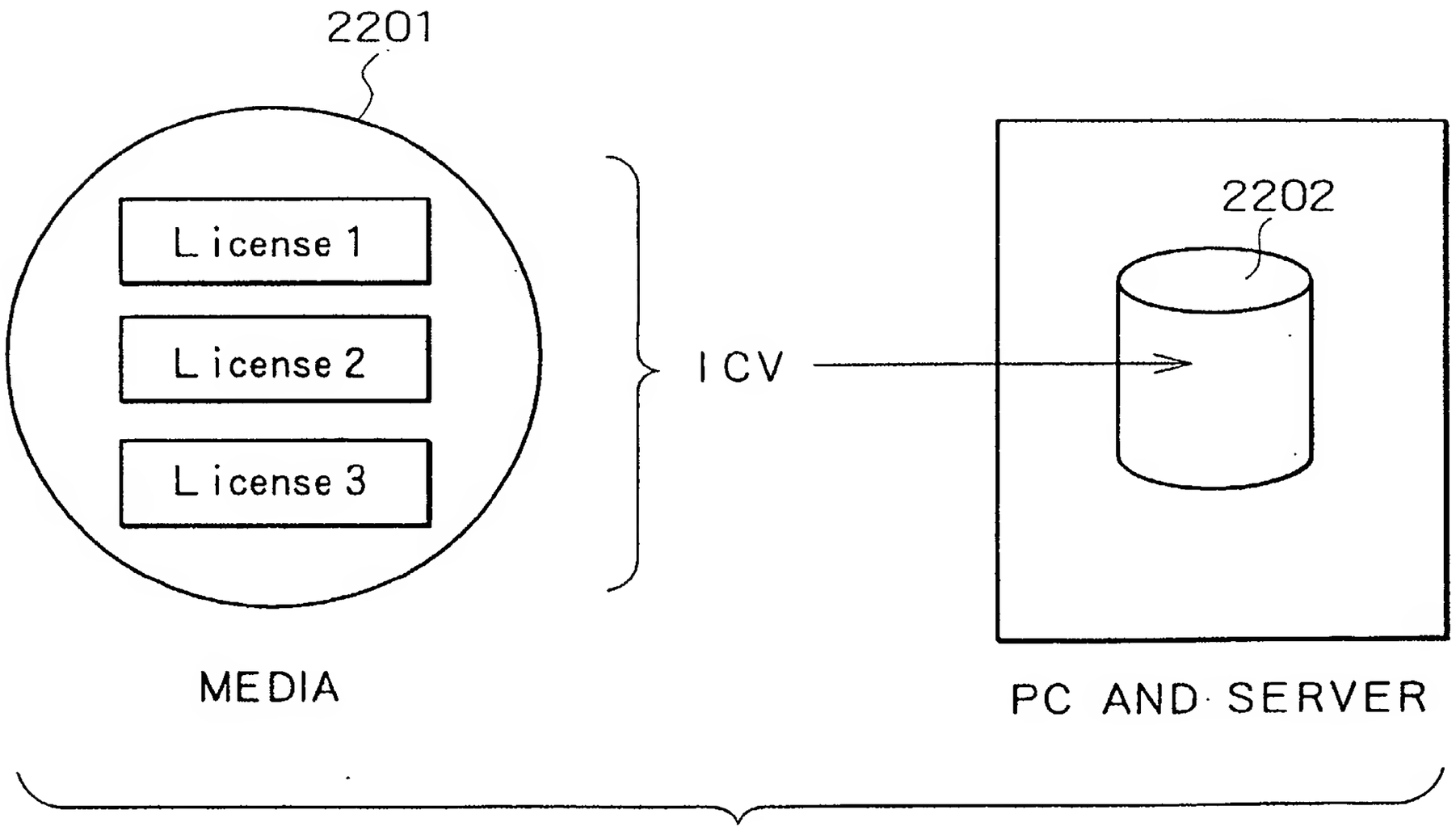


FIG. 49B

FIG.50


2201
License 1
License 2
License 3
MEDIA
ICV
2202
PC AND SERVER